University of Alberta

Library Release Form

Name of Author: Ki-Hwan Kwon

Title of Thesis: Hand Pose Recovery with a Single Video Camera

Degree: Master of Science

Year this Degree Granted: 2001

To infinity and beyond!
- Toy Story

University of Alberta

Hand Pose Recovery with a Single Video Camera

by

Ki-Hwan Kwon

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2001

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Hand Pose Recovery with a Single Video Camera** submitted by Ki-Hwan Kwon in partial fulfillment of the requirements for the degree of **Master of Science**.

Abstract

Two strategies for determining 3D hand poses are proposed in this thesis. The main purpose of this research was to develop a hand pose algorithm suitable for real-time applications based on the Perspective-n-Point problem (PnP).

In Strategy 1, the palm pose is determined first with a real-time PnP pose algorithm, and then joint angles are computed through a set of closed-form equations we derived. Strategy 2 involves using a heuristic search which combines a real-time PnP pose algorithm with the minimization of a 1D function. Both algorithms were possible due to the modeling of the human hand with reduced degrees of freedom (DOF), without sacrificing too much of the hand's usefulness.

The complete hand-tracking system was implemented and tested with both Strategy 1 and Strategy 2. It continuously determines 3D hand poses from a stream of single monochrome images with a speed of up to 12 Hz for Strategy 1 and up to 8 Hz for Strategy 2.

Acknowledgements

I'd like to thank Dr. Hong Zhang, my supervisor, for all his guidance, and *patience*, especially for accepting me as a thesis-based student, providing me with an opportunity to do a research in Computer Vision and Robotics.

I'd like to thank Dr. Fadi Dornaika for his kind advice with expertise in Object Pose and useful programs.

I'd like to thank the members of my committee, Dr. Martin Jägersand, and Dr. Max Meng from the electrical engineering department, for their valuable comments on my research.

I'd like to thank Dr. YiSheng Guan for helping me understand basic concepts in Robotics.

I'd like to thank Steve Sutphen for his good explanations in hardware, Dana Cobzas for many talks about cold weather in Edmonton and not having windows in Robotics Lab, and Andrzej Zadorozny for Smarties and other interesting stuff.

I'd like to thank my parents and parents-in-law for their spiritual and *financial support*. I'd like to thank my wife for her support and ideas, especially for her *craft tools* which were essential for making my research tools. I also thank my daughter GulSam and my son Noah for giving me *the joy of life*.

Contents

List of Figures

Chapter 1

Introduction

We define *hand pose recovery* as determining the position and orientation of the human palm in 3D space, and the joint angles of the fingers. People also use *hand tracking* to mean 3D hand pose recovery. Hand tracking is different from gesture recognition since hand tracking is concerned with three dimensional hand pose analysis while gesture recognition is the mapping from image sequences to a set of discrete classes and is usually done in 2D [4]. Hand tracking is more general and can be applied to many applications including gesture recognition.

Some of the important applications of hand tracking are the tele-operation of a robot hand, human computer interface, realistic hand animation in computer graphics, and interaction with the environment in virtual reality. It is difficult to control a multi-fingered robot hand autonomously because of its many degrees of freedom (DOF). The algorithms for the autonomous control of such a hand are complex, while tele-operation of a hand is relatively inexpensive and simple [14]. Naturally, with many DOFs, a human hand can be the best master for the tele-operation of a slave robot hand. In computer graphics, realistic animation of the human hand is not an easy problem. One approach is obtaining the motion parameters by tracking real human motion. In everyday life, we use gestures when we communicate with each other. If computers can understand human gestures, the communication between humans and computers will be more natural and faster. In virtual reality, the user has to deal with a 3D environment. Instrumented gloves are often used to facilitate the user interacting with this virtual environment.

Tracking the motion of articulated objects, especially the motion of the human body, has been a frequent topic of computer vision research ([19], [20], [21], [22], [23]) because the human body has a well-defined structure which can be converted

into a model . The human hand is similar to the body in that the hand has fingers, and the human body has arms and legs. [4] presents three difficulties in tracking articulated objects such as human limbs or the hand: the large number of degrees of freedom involved in their motion, non-linearities in the mapping from the DOF's to the image motion, and the presence of complex occlusion effects.

There are several approaches to recovering hand poses in 3D space. One popular method is to use an instrumented glove capable of sensing finger joint angles and its own position in 3D space. The main problem with an instrumented glove is its invasiveness since the user has to wear a glove wired to a machine. Also, safety problems can arise in situations in which the user has to put on some device such as a head-mounted display and can't see the real environment. In contrast, computer vision based approaches have been attempted but with limited success. The biggest problem with computer vision based approaches is the occlusion problem arising when fingers or markers on the fingers are hidden from the camera. No general solution has been found that allows real freedom for hand motion. Some studies demonstrated the success of their algorithms in the case of partial occlusion [4]. As well, computer vision based approaches are slow due to their complex algorithms and costly image processing. For a computer vision based hand pose recovery system to be successful and widely available, four essential elements should be satisfied: It should be inexpensive, fast, robust, and accurate. To be inexpensive, the system should be simple. Rather than using several cameras, one is preferred. The system should be fast since for most hand-tracking applications (e.g., gesture recognition) real-time performance is one of the most desirable components. It should be robust in dealing with occlusion and image noise. Of course, the computed hand pose should be accurate.

In this thesis, we describe two new algorithms to determine 3D hand poses. The major motivation was to apply the Perspective-n-Point (PnP) problem [8] to hand pose recovery. If we have three or more points attached to an object and we know the geometry among these points, we can determine the pose of the object in 3D space. If we do not consider the skin deformation of the hand, the hand can be regarded as an articulated object with many rigid segments. This assumption gives us the possibility to determine a hand pose with PnP algorithms. There are several advantages in using PnP to recover 3D hand poses. First, since we use points or simple markers representing these points, feature extraction will be easy,

reducing image processing costs. Second, there are many PnP algorithms already available including closed-form solutions. As a result, we may be able to recover 3D hand poses using a set of closed-form equations or some fast linear PnP algorithms. Third, a single camera can be used. These advantages provide a good possibility for a practical real-time hand-tracking system.

1.1 Towards applying the Perspective-n-Point problem to hand tracking

1.1.1 The Perspective-n-Point problem

In computer vision, an important problem is to determine the pose (how far away the object is and how much the object is rotated with respect to the camera) of a rigid object in 3D space. One special category of the problem is called the Perspective-n-Point problem [8], in which one tries to determine the pose of an object using points and their geometry. If we do not know the geometry of the points on the object, there can be infinite 3D interpretations of single 2D images. There are closed-form and numerical solutions available for this PnP problem (see Chapter 4 for references and a brief discussion).

1.1.2 Using points

In PnP, we use points which can be represented by simple markers such as circular blobs. It is easy to build and use a hand model comprised of points, compared to those of other complex features such as templates, edges, and lines. A basis for using points for a model can be found in psychological experiments on the human motion perception. These experiments [7] show that humans can recognize a subject's motion in 3D space by observing only points (actually light markers) attached to the subject. Dorner [3] described a hand model of points (centers of markers) and demonstrated success of the point model.

1.1.3 Rigid palm

We assume the palm is rigid without considering palm deformation. No studies in the hand-tracking literature analyzed palm deformation in their algorithms, although palm deformation was allowed as noise. If the palm is considered rigid[1], the hand is an articulated object with all rigid segments. With a rigid palm, we can place

[1]Skin deformations caused by finger joint flexion are not considered, either.

simple markers such as points on the palm, and we can expect that they will not move independently of each other. This makes it possible to use the geometry of the points on the palm to determine the pose of the palm. Since in natural hand motion, the palm is deformed and its surface is moving, attaching simple points on the palm and keeping them in place is difficult. With the help of the some rigid material such as thin wood board, we can force the palm to be rigid. Points are put on the board, and the board can be attached to a glove to be worn by a human subject. In this way, even palm deformation will not cause the points to move independently.

1.1.4 No abduction and adduction

We had to assume that the bottom joints joining the fingers to the palm had only one DOF instead of two. With only one DOF in the bottom joints, abduction and adduction movements of the fingers are not possible. However, the hand still retains most of its usefulness. Our first goal was to derive a set of equations allowing for the reconstruction of a 3D hand pose based on PnP rather than finding the hand pose by fitting a hand model to extracted image features. We also tried to recover joint angles using as small a number of points as possible on each segment of the fingers since we could not put many points on a small finger segment, and human limb motion experiments [7] suggested that as small a number of points as two was sufficient to interpret the limb motion. To satisfy these goals, it was necessary in our case to reduce one DOF from the bottom joints. This reduction also makes a one-dimensional search possible in our second algorithm.

1.2 Hand pose algorithms

1.2.1 Strategy 1

In Strategy 1, we tried to derive a set of closed-form equations to determine a hand pose. However, doing so is difficult since the human hand has many rigid segments. At least three points are required to determine the pose of a rigid object in 3D space. Fischler and Bolles [8] showed that even with three points at most four solutions could be found. This is why Dorner [3] stated, "Hence we cannot use the marker scheme to simply reconstruct the 3D hand shape segment by segment".

However, if we note that the segments of the hand are kinematically dependent, we may not need three or more points on all the rigid segments of the hand. In fact, only one segment would require multiple points, and for all the other segments, only

one point would be sufficient to determine a hand pose if these are considered with respect to the one rigid segment with multiple points. Strategy 1 was developed in this context. The palm pose is determined from the four points on the palm with a PnP pose algorithm. Each joint angle is determined with one point on each finger segment. This is possible because the palm pose is determined before the finger joint angles are computed.

1.2.2 Strategy 2

In Strategy 2, each joint angle is found with a 1D heuristic search. During the search, a corresponding joint of the hand model is rotated from a lower bound to an upper bound incrementally. At each rotation, how close the current angle is to the actual angle is evaluated using a PnP pose algorithm.

The range of the heuristic search for a joint angle is limited roughly to the physical limits of the actual finger joints at the beginning. Once the initial joint angles are determined, the search space can be considerably reduced assuming the frame rate is high. Since the evaluation step is done with a PnP pose algorithm, it is important to choose a fast PnP pose algorithm to make the system reasonably fast. Strategy 2 is more stable than Strategy 1.

1.3 The complete hand-tracking system

A complete hand-tracking system was implemented and tried using both strategies. The complete system with Strategy 1 is depicted in Figure 1.2. In box A, an input image is captured with a single camera. Once an image is obtained, markers are extracted, and their centroids are computed (box B). Marker correspondence is established in this step based on marker size and hand geometry. With marker correspondence, the palm pose is determined, and closed-form equations are used to compute joint angles (box C). There are four possible angles obtained for each joint angle, and an angle close to the actual one should be chosen. This is done as follows. We project the corresponding finger with the four possible angles (box E). We compute the residual error between the actual image point of the finger marker and four projected ones, and then we choose the angle with the least residual error (box F). The bottom joint angle of each finger should be computed prior to computation of its middle joint angle. Once we find all the joint angles, they are transferred to an OpenGL graphic hand model for visual feedback (box D).

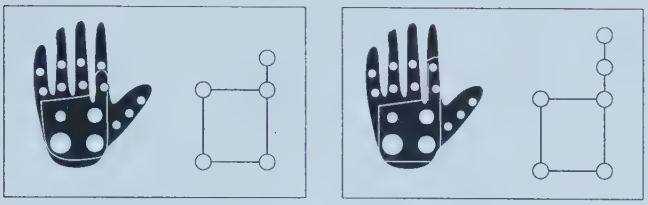


Figure 1.1: Markers and the hand model required to determine the index finger bottom joint angle (left box) and the middle joint angle (right box) are shown. Five markers are needed for the bottom joint, and six markers for the middle joint. This is true to all the other fingers.

Figure 1.3 describes the system when Strategy 2 is applied instead of Strategy 1, as shown in box C. With marker correspondence (box B), an heuristic search is performed for each joint to find the correct joint angle (box C). An heuristic search for a joint angle can be decomposed into four steps. These four iterative steps should be performed for each joint angle. In box C1, a subset of the complete hand model is selected according to which joint angle we want to compute (e.g., computation of the index finger bottom joint angle will require five markers and the corresponding part from the complete hand model, as shown in Figure 1.1). Initially, the model joint will be set to a lower bound and be incremented by one degree each time. A pose of the model is computed (box C2). With the pose computed (K) and the camera's intrinsic parameters, the model can be projected to the camera's image plane (box C3). In box C4, we compute the amount of residual error between the actual image points and the projected points. These four steps are repeated until an upper bound is reached. The angle with the least residual error is assumed to be an actual joint angle.

1.4 Thesis organization

The thesis is organized as follows:

- In Chapter 2 previous work related to this research is surveyed.
- Chapter 3 provides the basic concepts concerning camera modeling and calibration.
- In Chapter 4 definitions related to object pose are listed. Perspective-n-Point pose algorithms are surveyed, and the pose algorithm involved in our system is discussed in detail.

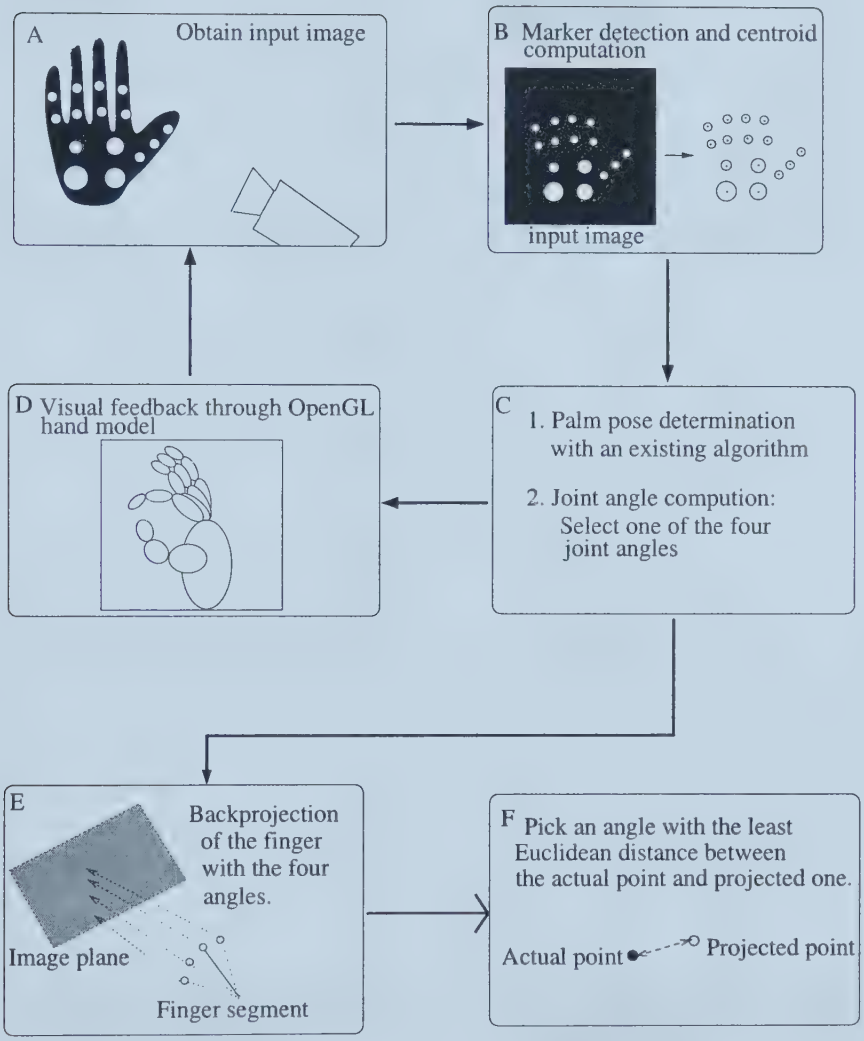


Figure 1.2: An overview of the hand-tracking system when implemented with Strategy 1.

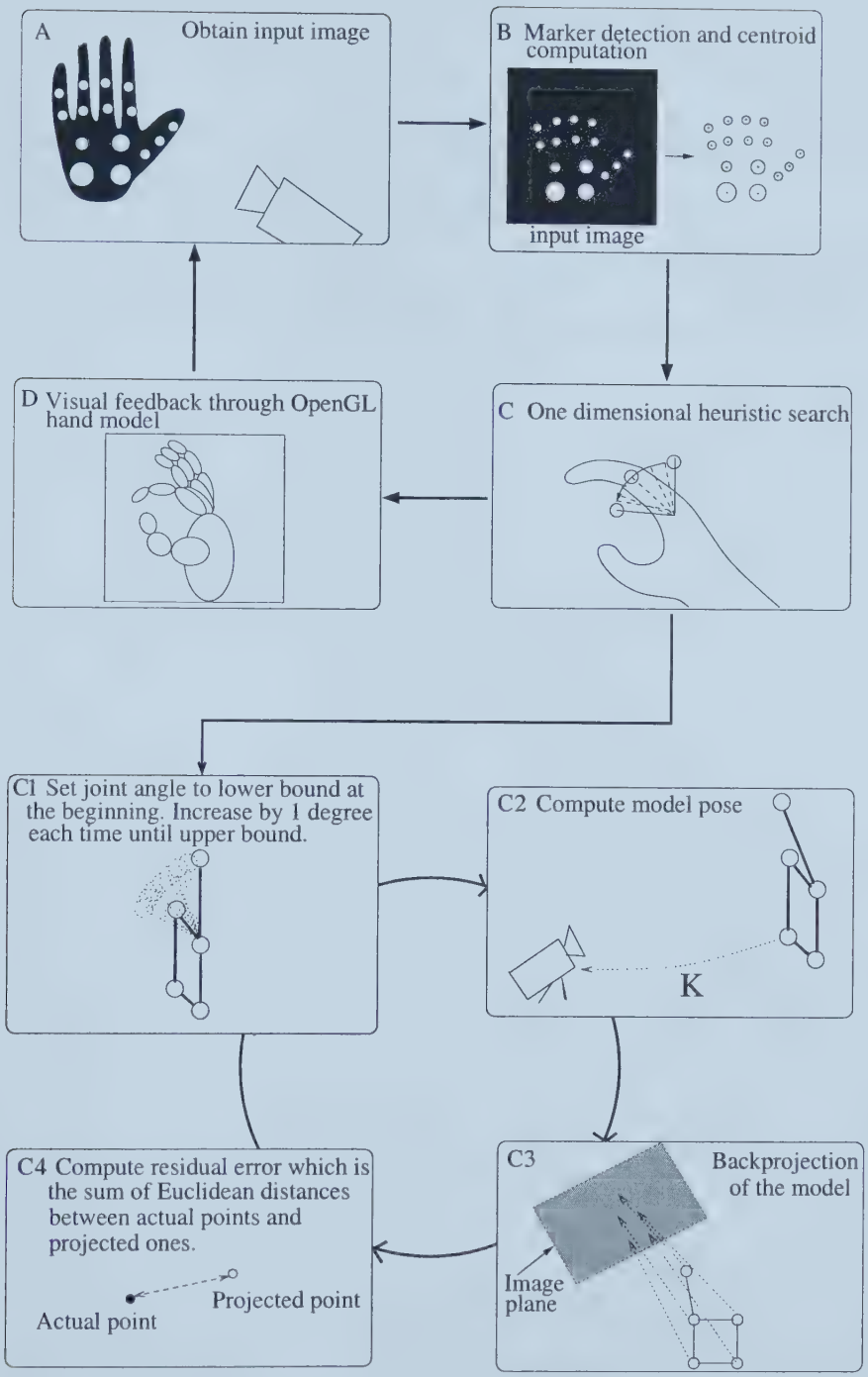


Figure 1.3: An overview of the hand-tracking system when implemented with Strategy 2.

- In Chapter 5 our hand model is compared to a real hand. Two new algorithms for recovering a hand pose in 3D space are elaborated on.
- Chapter 6 describes experiments on synthetic and real images to characterize the performance of the algorithms.
- Chapter 7 describes our complete system in detail. We show the success of our strategies by tracking the human hand in real time.
- In Chapter 8 a summary of this thesis is presented, and some future directions are suggested. Some applications of the current system are discussed.

Chapter 2

Previous work

In computer vision literature regarding tracking objects, there have generally been two classes of approaches: model-based and non-model based. In the non-model based approaches, the 3D structure and the pose of an object are recovered from the image sequences of the features, about which only general information is known. Since a high-level understanding of the scene is reached based on the results of the low-level processing, this is a bottom-up approach [20]. In the model-based approach, details about the model are assumed to be known, and this model guides the interpretation of the scene in the image (a top-down approach). In most of the work on human hand tracking, a model of the hand is employed to remove ambiguities and help interpret the image. Our work belongs to the model-based approach.

2.1 Tracking without a 3D model and object-specific knowledge

This approach is fundamentally different from our work since it does not employ a 3D object model. Rather, the 3D structure of an object is reconstructed using simple features observed for a sequence of frames. For example, Rashid [17] describes a system called "Lights" for interpreting an MLD (moving light display). The system can reconstruct the 3D structure of a walking man with thirteen points attached to his body. Doing so is possible because the motion of the related points on the walker provides some relationships in their 2D projection after a period of time. The projected positions and the projected velocity of the points are used to separate the points into groups which belong to different parts. Some of the studies in this line of research are [15], [26], and [27].

2.2 Tracking with a 3D model and object-specific knowledge

2.2.1 General human motion tracking

O'Rourk [20] used constraint propagation and high-level prediction to analyze human motion in 3D. His system was composed of four processes: image analysis, parsing, prediction, and simulation. A simulator was used to obtain the positions of a predicted model so that it could be interpreted. Constraint propagation was employed to limit locations of related body parts from the known location of a body part. All four steps were guided by a 3D human model.

Hogg [19] used a hierarchical Walker model to produce a description of a person walking in a TV scene. The system generated a large number of instances that could span all the possible descriptions of a walker and find the instance that best matched the walker in the image. Temporal and dynamic constraints were imposed to reduce the infinite search space.

2.2.2 Human hand tracking

Dorner [3] presented a hand-tracking system to serve as an interface for American sign language signers. The system used non-linear optimization techniques to recover 26 parameters of the hand by minimizing the difference between the projected model and the features in the image. She used a color encoded cotton glove. The system did not work in real time. Dorner's approach is similar to ours in that both systems make use of the position of the markers to determine the pose of a hand, and the user has to wear a cotton glove.

Rehg [4] implemented a system called DigitEyes which tracked an unadorned hand using line and point features. A projected model and features in the image were registered by minimizing a residual function. DigitEyes was implemented on a special hardware for real-time performance. It was the first real-time 3D hand-tracking system (up to 10 Hz). Stereo was employed to track a whole hand. The system needed to be initialized prior to tracking by requiring the user to place her hand at a known configuration.

Segen [5] implemented a system to drive an articulated hand model using inverse kinematics. The positions of the finger tips in the stereo images were found using a contour detection algorithm, and 3D positions of the finger tips were then deter-

mined. Joint angles were computed with inverse kinematics. His system worked in real time but it controlled only the wrist, index finger, and thumb. The system used stereo images. He showed that it could be used as a 3D input device.

Quentin [24] proposed a system which recovers a 3D hand pose using stereo depth images. The 3D hand model is fitted to a reconstructed 3D scene. His hand model is made of cones and spheres. The system required an initial pose to be known in the beginning.

Lee and Kunii [16] described a system for inferring the 3D structure of a hand using multiple views of seven characteristic points. Inverse kinematics were used to get correct finger angles. They exploited the joint constraint that the top joint angle of a finger is two thirds of the middle joint angle, which we also assumed for our complete system.

The advantages of our work compared to others' are as follows:

- The positions of simple markers are used for our hand model while [4] [24] use models made of lines, spheres, and cones. Making a simple hand model means fast image processing and easy projection.
- A single camera is used, where as [4] [24] use stereo rigs or multiple views ([16]), making the system cheaper.
- The position of the palm is determined by the palm markers, where as [4] [24] usually require the hand to be placed in a known position in the beginning.
- Our solutions include closed-form equations (Strategy 1) which compute joint angles directly from point correspondence, while [3] [4] [24] use iterative model fitting approaches.

Chapter 3

Camera modeling and calibration

For any computer vision system which depends on measurements through a camera, accurate camera calibration is an essential preliminary step. Without accurate camera parameters, algorithms fail to work in practice even if they are correct theoretically.

3.1 A pinhole camera model and calibration matrix

To understand image formation, usually a pinhole camera model is employed in computer vision although, in reality, lenses are used to focus an image onto the camera's focal plane.

The pinhole camera is the simplest and the ideal camera model. It has a pinhole through which light enters before forming an inverted image on the camera imaging plane behind the pinhole. In practice, we often model a pinhole camera by placing the virtual image plane between the pinhole of the camera and the object so that the image is not inverted. This is easier to work with since the image displayed on the computer screen is already a corrected (non-inverted) image. Also, it is mathematically equivalent to the one in which the image plane is behind the focal point (pinhole). The mapping of three dimensions onto the image plane is called *perspective projection* (see Figure 3.1).

A coordinate frame having its origin at the center of projection C and whose z-axis is along the optical axis is called the *standard camera coordinate frame*. The *optical axis* is the straight line passing through the center of the projection and perpendicular to the image plane. We denote by P_i a 3D point with coordinates

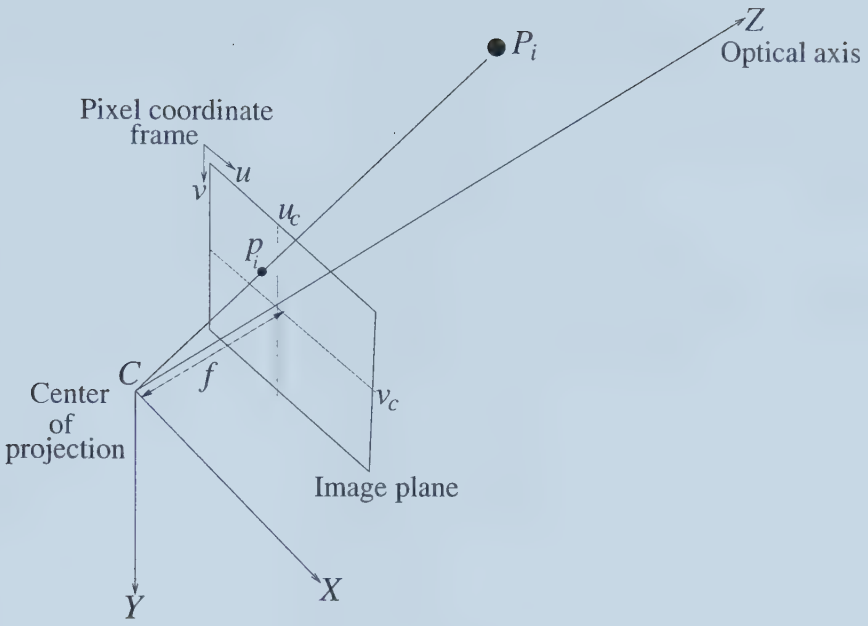


Figure 3.1: Pinhole camera model with virtual image plane in front of the pinhole.

X_i, Y_i, and Z_i in the camera coordinate frame. p_i is a point lying on the image plane with coordinates x_i, y_i and f in the camera frame. From similar triangles, the relationship between the two points P_i and p_i is given by

$$x_i = \frac{X_i f}{Z_i} \tag{3.1}$$

$$y_i = \frac{Y_i f}{Z_i} \tag{3.2}$$

These are non-linear equations and can be written linearly using homogeneous coordinates as follows:

$$\begin{bmatrix} sx_i \\ sy_i \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

where s is a non-zero scale factor.

Note that x_i and y_i are so far expressed in the unit based on the camera frame, whether it is millimeter or meter. Since we are dealing with digital images whose coordinates are expressed in pixels, it is convenient to express (x_i, y_i) in pixels. Usually, the origin of the *pixel coordinate frame* is located at the top left hand corner of the image plane (Figure 3.1) with axis u and v. The actual pixel coordinates of p_i are given by

$$u_i = u_c + \frac{x_i}{dx} \tag{3.3}$$

$$v_i = v_c + \frac{y_i}{dy} \tag{3.4}$$

where dx is pixel width and dy pixel height in a sensor array.

If we substitute Equations (3.1) and (3.2) into (3.3) and (3.4) respectively and multiply through by Z_i, the following expressions are obtained:

$$Z_i u_i = Z_i u_c + \frac{X_i f}{dx}$$

$$Z_i v_i = Z_i v_c + \frac{Y_i f}{dy}$$

i.e.,

$$\begin{bmatrix} su_i \\ sv_i \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

where s is the scale factor with the value equal to Z_i, $\alpha_u = \frac{f}{dx}$ and $\alpha_v = \frac{f}{dy}$, and u_c and v_c are the pixel coordinates of the optical axis at the image plane.

We can write the above equation in a simpler form:

$$\mathbf{p} = \mathbf{M} \cdot \mathbf{P} \tag{3.5}$$

where $\mathbf{p}$ is a vector of homogeneous pixel coordinates and $\mathbf{P}$ is a vector of the homogeneous world coordinates of a 3D point with respect to the camera frame. We call the matrix $\mathbf{M}$ the *perspective projection matrix*. Since α_u, α_v, u_c and v_c do not change with the position and orientation of the camera, they are called *camera intrinsic parameters*.

In practice, we do not specify world points based on the camera frame. Usually, we introduce a world frame, and the world points are expressed in this frame (see Figure 3.2). In Figure 3.2, the coordinates of a 3D point P_i are specified in a world frame which does not coincide with the camera frame. $\mathbf{P_i}$ is the vector from P_0 to the point P_i. Assume $\mathbf{K}$ is the 3D transformation from the world frame to the camera frame defined as

$$\mathbf{K} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{i}^T & & t_x \\ & \mathbf{j}^T & & t_y \\ & \mathbf{k}^T & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

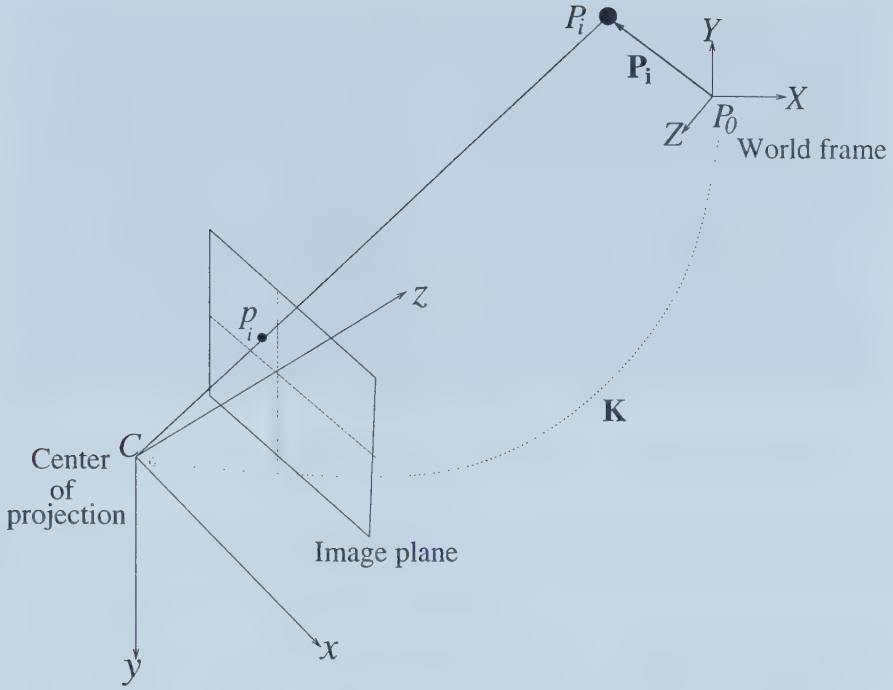


Figure 3.2: A point P_i is defined in a world frame with the origin at P_0. This world frame does not coincide with the camera frame.

$\mathbf{K} \cdot \mathbf{P_i}$ will give the coordinates of $\mathbf{P_i}$ in the camera frame. Now, Equations (3.1) and (3.2) are expressed as follows:

$$x_i = f \frac{\mathbf{P_i} \cdot \mathbf{i} + t_x}{\mathbf{P_i} \cdot \mathbf{k} + t_z} \tag{3.6}$$

$$y_i = f \frac{\mathbf{P_i} \cdot \mathbf{j} + t_y}{\mathbf{P_i} \cdot \mathbf{k} + t_z} \tag{3.7}$$

By considering camera intrinsic parameters, we can express the pixel coordinates of (x_i, y_i) as follows:

$$u_i = \alpha_u \cdot \frac{\mathbf{P_i} \cdot \mathbf{i} + t_x}{\mathbf{P_i} \cdot \mathbf{k} + t_z} + u_c \tag{3.8}$$

$$v_i = \alpha_v \cdot \frac{\mathbf{P_i} \cdot \mathbf{j} + t_y}{\mathbf{P_i} \cdot \mathbf{k} + t_z} + v_c \tag{3.9}$$

Equation (3.5) can be rewritten as follows when $\mathbf{P}$ is a vector of homogeneous world coordinates of a 3D point which are specified with respect to a world frame:

$$\mathbf{p} = \mathbf{M} \cdot \mathbf{K} \cdot \mathbf{P}$$

These rotation and translation parameters in $\mathbf{K}$ are known as *extrinsic camera*

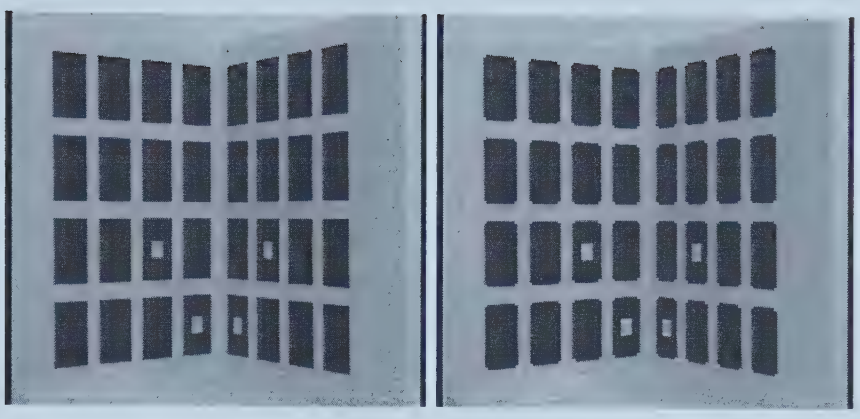


Figure 3.3: The calibration pattern and its detected corners.

parameters. The projection matrix **M** and transformation matrix **K** combine to form a single 3×4 *camera calibration matrix* **C**:

$$\mathbf{C} = \begin{bmatrix} \alpha_u r_{11} + u_c r_{31} & \alpha_u r_{12} + u_c r_{32} & \alpha_u r_{13} + u_c r_{33} & \alpha_u t_x + u_c t_z \\ \alpha_v r_{21} + v_c r_{31} & \alpha_v r_{22} + v_c r_{32} & \alpha_v r_{23} + v_c r_{33} & \alpha_v t_y + v_c t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

3.2 Actual camera calibration

Camera calibration is the process of finding accurate camera intrinsic and extrinsic parameters. Faugeras's [25] and Tsai's [11] algorithms were used, and we chose the better result between them. Both algorithms assume the pinhole camera model. Tsai's method also provides the radial distortion parameter of the camera, but in our case, the camera had negligible distortion with its narrow field of view.

Camera calibration algorithms require the world coordinates of the features (the corners of the rectangles in our case) in a calibration pattern and the corresponding pixel coordinates of its image. A coordinate frame was attached to the calibration pattern, and then, the world coordinates of each corner were determined. The corresponding pixel coordinates of each corner of the rectangles were extracted from an image of the calibration pattern (see Figure 3.3).

We made an L shaped frame onto which laser-printed calibration patterns were attached. The calibration pattern and the corner extraction program used were designed by [31]. The calibration program for Tsai's method is available at [30].

Chapter 4

Object pose

In Strategy 1, an existing object pose algorithm is used in determining the position and orientation of the palm in 3D space. In Strategy 2, this object pose algorithm is applied repeatedly to compute residual errors to find a correct joint angle. Among many object pose algorithms, we exploited the weak perspective algorithm developed by [1] and [29]. If speed is not an important issue, any other PnP object pose algorithm can be used in our hand-tracking system.

4.1 Terminology

Consider a simple setup as depicted in Figure 4.1. A coordinate frame attached to an object is referred to as *object frame*. The object frame is attached to a box, and one of the feature points is located at the origin O of the object frame. The feature points on the object may be referred to as *object points*. The coordinate frame with origin C is the standard camera coordinate system as mentioned in the previous chapter. The position and orientation of the object frame with respect to the camera frame are called *object pose*. In Figure 4.1, the position of the object is expressed in a translation vector $\mathbf{t}$, which is the position of the origin of the object frame with respect to the camera frame. Orientation is a rotation matrix whose rows are the coordinates of the unit vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ of the camera frame expressed in the object coordinate system. This translation and orientation make a transformation matrix which represents an object pose.

4.2 An overview of existing PnP algorithms

Determining an object pose using images of feature points with their known geometry is a well-known problem in photogrammetry and computer vision. This so-called

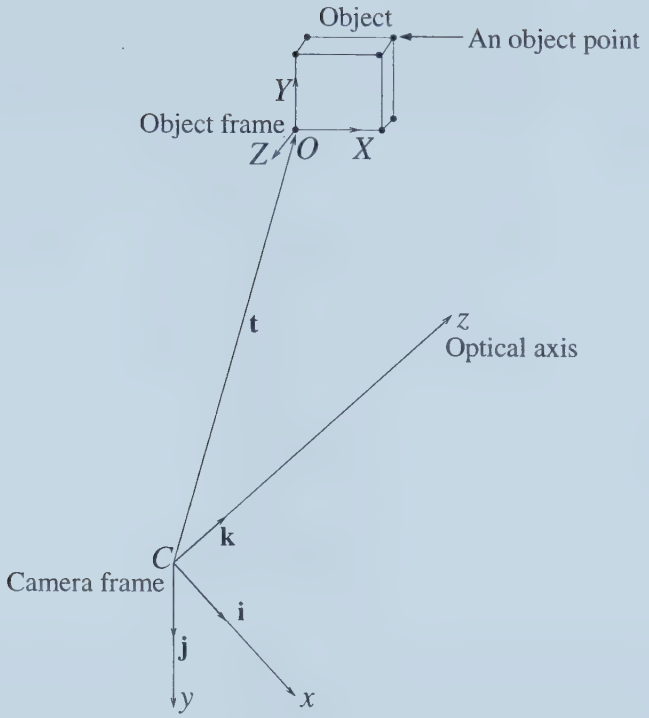


Figure 4.1: General setup for an object pose. A coordinate frame is attached to an object, and the feature points on the object are defined in this coordinate frame. An object pose is defined as the orientation and translation of this coordinate frame with respect to the camera frame.

Perspective-n-Point problem [8] can be solved with closed-form or numerical solutions. Closed-form solutions have been applied to only a limited number of points ([8], [9]). Numerical solutions ([10], [13]) were required for an arbitrary number of points. However, good initial guesses were necessary in order for the numerical algorithms to converge. They were also computationally expensive. Recently, a linear n-point algorithm was suggested by [12], making a closed-form solution available with at least four or more points.

4.3 The POSIT algorithm

We used an iterative method called POSIT ([29], [1]) for our hand-tracking system. The algorithm exploits scaled orthographic projection (SOP). In SOP, object points are projected onto a plane parallel to the image plane and passing through the origin of the object frame, after which they are projected with the true perspective onto the image plane (Figure 4.2). The algorithm has several advantages compared to non-linear methods. First, it does not require initial guesses. Second, it is fast and

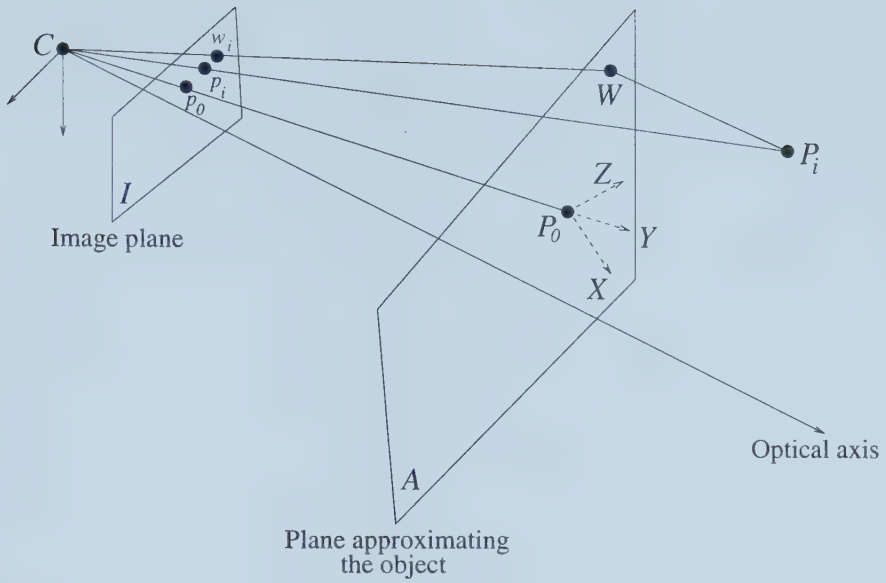


Figure 4.2: Perspective projection (p_i) and scaled orthographic projection (w_i) for an object point P_i. For scaled orthographic projection, P_i is first projected onto W on the plane A using orthographic projection, and then the point W is projected onto the image plane I using the perspective projection. The reference point P_0 is projected onto p_0 in both projections. Plane A is parallel to image plane I and passes through the origin P_0 of the object frame.

so is suited to real-time applications. Like non-linear methods, it can be applied to an arbitrary number of points. Now, a brief summary of the algorithm is in order.

As we saw in Chapter 3, a 3D point in an object frame, P_i, and its image coordinates (x_i, y_i) have the following relationship in the pinhole camera model ($\mathbf{P_i}$ is the vector from the origin P_0 of the object frame to the point P_i):

$$x_i = f\frac{\mathbf{P_i}\cdot\mathbf{i} + t_x}{\mathbf{P_i}\cdot\mathbf{k} + t_z} \tag{4.1}$$

$$y_i = f\frac{\mathbf{P_i}\cdot\mathbf{j} + t_y}{\mathbf{P_i}\cdot\mathbf{k} + t_z} \tag{4.2}$$

when the 3D transformation from the object frame to the camera frame is defined as

$$\begin{bmatrix} & \mathbf{i}^T & & t_x \\ & \mathbf{j}^T & & t_y \\ & \mathbf{k}^T & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By dividing both terms of the fraction of Equations (4.1) and (4.2) by t_z, we have the following equations:

$$x_i(1+\epsilon_i) - x_0 = \mathbf{I} \cdot \mathbf{P_i} \tag{4.3}$$

$$y_i(1+\epsilon_i) - y_0 = \mathbf{J} \cdot \mathbf{P_i} \tag{4.4}$$

where

$$\mathbf{I} = f\frac{\mathbf{i}}{t_z},\ \mathbf{J} = f\frac{\mathbf{j}}{t_z}$$

$$x_0 = f\frac{t_x}{t_z},\ y_0 = f\frac{t_y}{t_z}$$

$$\epsilon_i = \mathbf{k} \cdot \mathbf{P_i}/t_z$$

$$\mathbf{k} = \mathbf{i} \times \mathbf{j}$$

Note that x_0 and y_0 are known because they are the image coordinates of the origin P_0 of the object frame.

The basic idea behind the algorithm is that whenever ϵ_i is fixed to *any* value, Equations (4.3) and (4.4) become linear in $\mathbf{I}$ and $\mathbf{J}$. Finding an object pose with a fixed ϵ_i actually amounts to finding the pose with the image points projected with the *scaled orthographic projection* because $x_i(1+\epsilon_i)$ and $y_i(1+\epsilon_i)$ are actually the SOP image coordinates of an object point whose image coordinates of the perspective projection are x_i and y_i. Consider Figure 4.3. w_i is the SOP projection of the object point P_i, and p_i is the perspective projection of the same object point. The coordinates (x_i', y_i') of w_i are related to the coordinates (x_i, y_i) of p_i by

$$x_i' = x_i(1+\epsilon_i)$$
$$y_i' = y_i(1+\epsilon_i)$$

Since we do not know the exact value of ϵ_i, we have to set it to a fixed value. The simplest setup is to set $\epsilon_i = 0$. Note that ϵ_i will be small when the object is at some distance from the camera, i.e., when t_z is large (see the definition of ϵ_i above). $\epsilon_i = 0$ corresponds to assuming that $w_i = p_i$ and that the object point P_i is positioned at P_i'. By solving Equations (4.3) and (4.4), we obtain the approximate pose. Since ϵ_i is not the exact value, the pose computed is an initial approximation. This approximated pose is improved iteratively as follows:

1. For all $i \in \{1 \ldots n\} \geq 3$, $\epsilon_i = 0$.

2. Estimates of vectors **I** and **J** can be obtained by solving the linear system of Equations (4.3) and (4.4).

3. Compute the position and orientation of the object frame with respect to the camera frame by

   - Compute the scale s, i.e., $\frac{f}{t_z}$: $s = \| \mathbf{I} \|$, or $s = \| \mathbf{J} \|$, or average of the two
   - Compute the unit vectors: $\mathbf{i} = \frac{\mathbf{I}}{s}$, $\mathbf{j} = \frac{\mathbf{J}}{s}$, $\mathbf{k} = \mathbf{i} \times \mathbf{j}$
   - Compute the translation: $t_z = f/s$, $t_x = x_0/s$, $t_y = y_0/s$

4. For all i, compute:

$$\epsilon_i = \frac{\mathbf{k} \cdot \mathbf{P_i}}{t_z}$$

If the computed ϵ_i is the same as the previous one within a threshold, stop the iteration; otherwise, go on to Step 2 with the new ϵ_i. Each time a more accurate ϵ_i is computed, more accurate scaled orthographic projections of the object points are obtained.

The above algorithm works if the distance of the object from the camera is three or four times the size of the object. If the object is too far away, iterations do not improve the first result since the SOP and perspective projections are not different.

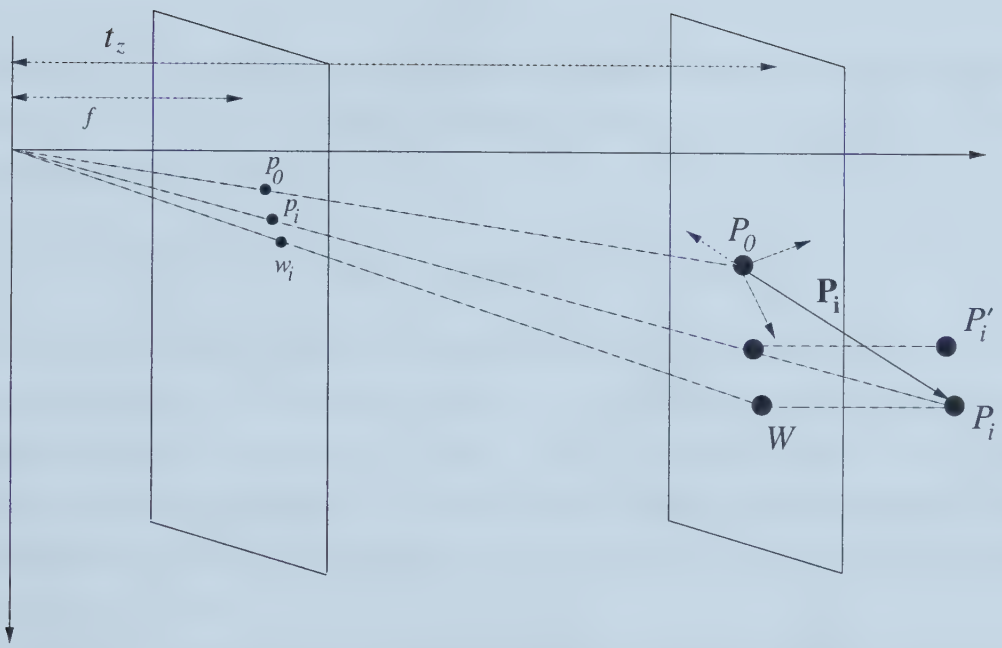


Figure 4.3: The coordinates (x_i', y_i') of w_i are $(x_i(1+\epsilon_i), y_i(1+\epsilon_i))$ in which x_i and y_i are the coordinates of p_i. Setting ϵ_i to zero is equal to assuming that $w_i = p_i$ and that the object point P_i is positioned at P_i'. w_i will be computed more accurately in each iteration and move to the correct position, resulting in the computation of the accurate object pose. Adapted from Figures 1 and 2 in [1].

Chapter 5

Hand pose

In this chapter, we describe our hand model, which is a simplified version of the real human hand. Two new algorithms (Strategies 1 and 2) for computation of a hand pose are described in detail.

5.1 The human hand vs. our hand model

The human hand has a complex structure with 23 DOF's (Figure 5.1). Six DOF's are added when the hand moves in 3D space. Each finger has four DOF's. Metacarpophalangeal (MCP) joints of each finger have two degrees of freedom, one of them being used for moving fingers in the palm plane (abduction and adduction). The thumb has five DOF's. The thumb's dexterity comes from the Trapeziometacarpal joint with three DOF's.

However, it is difficult to model the human hand as it is biologically, and it is not necessary to do so for a model to be useful for some practical applications. Usually the human palm is regarded as rigid in hand tracking based on vision techniques. The thumb is simplified, and DOF's of the fingers are reduced. Instrumented gloves cannot recover all the DOF's of the thumb either [32].

In our model (Figure 5.2), distal interphalangeal (DIP) joints are assumed to be completely dependent on proximal interphalangeal (PIP) joints based on the fact that a person cannot move the DIP joint of a finger without moving the PIP joint of the finger in natural hand motion. Rijpkema and Girard [2] suggested that the relationship between the DIP and the PIP joint angles is almost linear. This coupling is not necessary for our algorithms to work, and our current algorithms can handle more joints as long as the joints have one DOF as the DIP joint does. However, this assumption of coupling will increase the speed of our hand-tracking

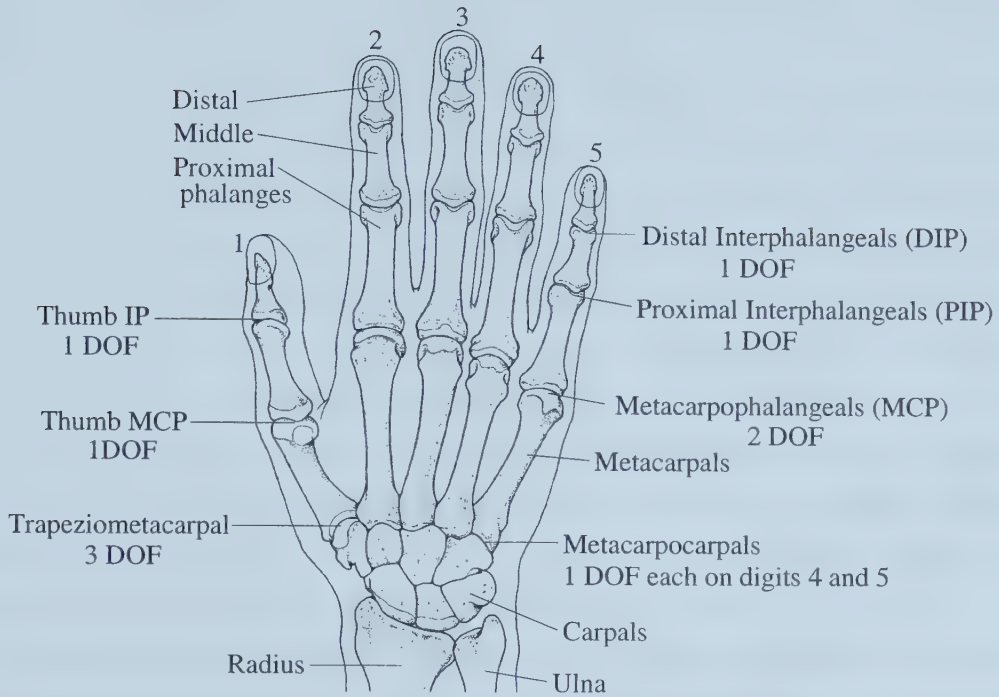


Figure 5.1: Real hand with 23 DOF's. Adapted from [18].

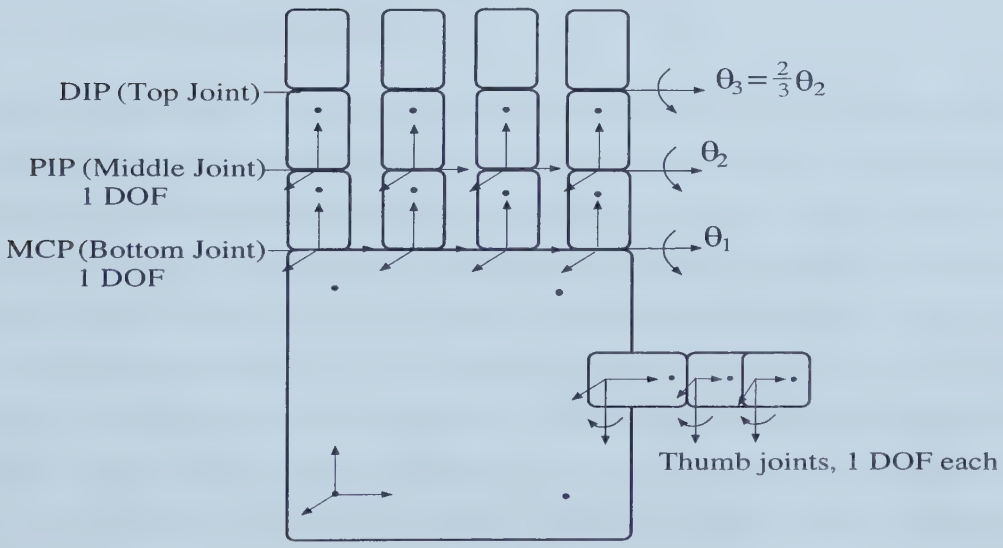


Figure 5.2: Our hand model. The DIP (top) joints are dependent on the PIP (middle) joints. The MCP (bottom) joints have only one DOF. Abduction movements, therefore, are not considered. Each finger has only two DOF's. The thumb has three DOF's while the real hand thumb has five DOF's. The configuration of the fifteen feature points are also shown. The origin of the palm coordinate frame is attached to a point in the left bottom corner of the palm.

system, and we would like to use the known fact accepting small error that might be caused by this assumption. We consider that the relationship between the PIP joint and the DIP joint of a finger is linear and that the DIP joint angle is computed as two thirds of the PIP joint angle as suggested by [2]. The MCP joints of a real hand have two DOF's as mentioned in the beginning. However, for the algorithmic reasons mentioned in Chapter 1, the MCP joints of all the fingers in our model have only one DOF, making the fingers move only perpendicularly to the palm plane. Therefore, there are only two DOF's in each finger since the DIP joint is dependent on the PIP joint, and the MCP joint has only one DOF. With five DOF's, it is difficult to model a real thumb. In our case, to make the algorithms we developed for the four fingers work for the thumb as well, the thumb is modeled as having three links with one DOF for each joint. As mentioned in Chapter 1, the palm is modeled as being rigid, assuming the carpals have no DOF.

We model the human hand in this way so that we can come up with simple real-time hand-tracking algorithms without sacrificing too much of the hand's usefulness. We believe that there are still many applications for the simplified hand model once our hand tracking system is working since it still has 17 DOF's available.

5.2 Problem definition

Since we want to apply PnP to hand pose determination, we use points to model the hand. We attach four coplanar points on the palm. Four points are the minimum number required by the PnP pose algorithm that we use [1]. We try to use only one feature point for each segment of the fingers. Since it is assumed that the DIP joint is linearly related to the PIP joint, computation of the DIP joint angle does not require any marker. So the 17 DOF's (six DOF's for the palm, eight DOF's for the four fingers, and three for the thumb) of the human hand are determined by 15 points. Figure 5.2 shows the configuration of the feature points on the right hand.

Our problem can be stated as follows. Given one captured image featuring all the markers and the knowledge of the palm model and the fingers' kinematics, we estimate the palm pose as well as the joint angles of each finger (see Figure 5.3).

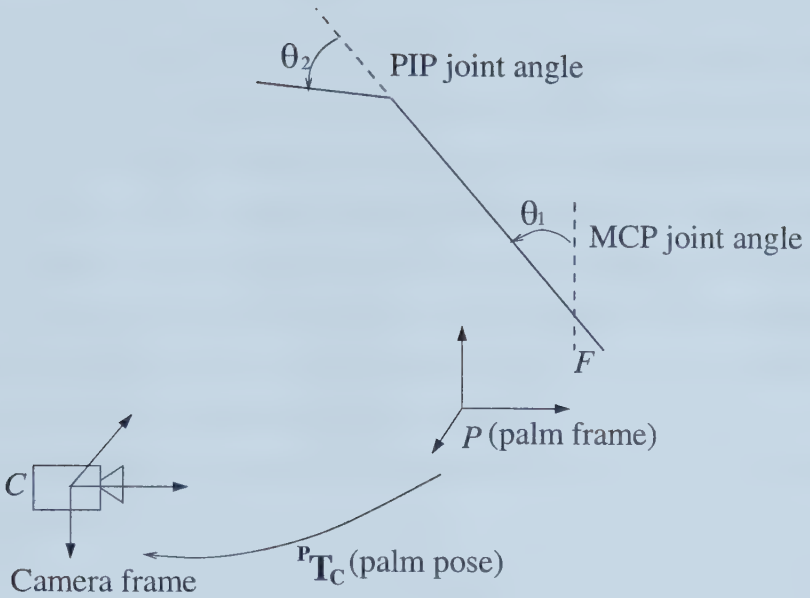


Figure 5.3: Problem definition of the hand pose (one finger is shown for the sake of simplicity). The position and orientation of the palm from the camera frame (${}^{\mathbf{P}}\mathbf{T}_{\mathbf{C}}$) should be determined, and for each finger, the MCP and PIP joint angles should be computed. The DIP joint angles are computed from the PIP joint angles.

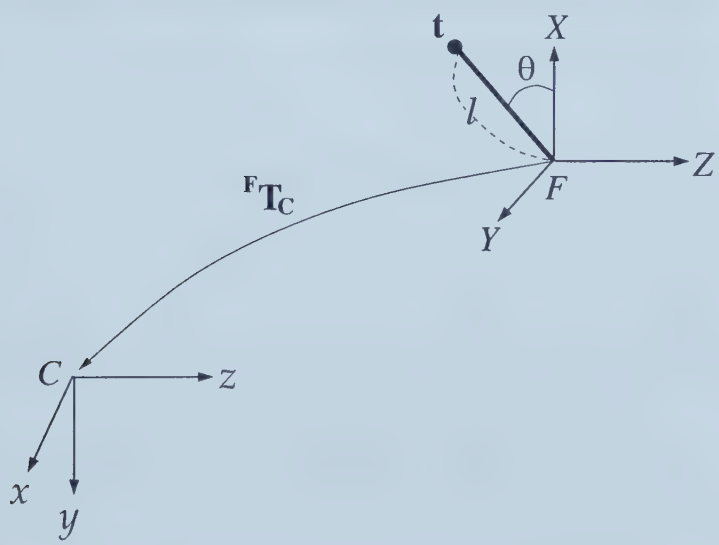


Figure 5.4: A simple setup for Strategy 1. A finger segment with length l is moving in the XY plane, rotating about the Z-axis. ${}^{\mathbf{F}}\mathbf{T}_{\mathbf{C}}$ combines all the 3D transformations including ${}^{\mathbf{P}}\mathbf{T}_{\mathbf{C}}$ (see Figure 5.3) required to make a transformation from finger frame F to the camera frame C. We consider only one joint angle in this setup since other angles can be recovered by applying the same algorithm repeatedly.

5.3 The hand pose algorithm: Strategy 1

Consider a simple setup as depicted in Figure 5.4. We consider a finger segment with only one joint angle in this setup because we assume that if we know how to compute one joint angle, other joint angles can be recovered by applying the same technique repeatedly. Also, in this setup, we are not concerned with the computation of the palm pose. Because our palm model has four feature points on it and their geometry is known, we can determine the palm pose with an existing PnP pose algorithm. So the hand pose problem is reduced to computing one joint angle. Therefore, we need to describe only how to determine one joint angle.

The coordinates of the finger tip $\mathbf{t}$ with respect to F in Figure 5.4 can be expressed as follows:

$$\mathbf{t} = \begin{bmatrix} l\cos(\theta) \\ l\sin(\theta) \\ 0 \\ 1 \end{bmatrix}$$

Note that the finger moves in the XY plane because it has only one DOF and hence is confined to a plane as in our hand model.

If we know the transformation ${}^{\mathbf{F}}\mathbf{T}_{\mathbf{C}}$, we can determine the pixel coordinates $\mathbf{p}$ of finger tip $\mathbf{t}$ using camera perspective projection matrix $\mathbf{M}$ (see Chapter 3) as follows:

$$\mathbf{p} = \mathbf{M} \cdot {}^{\mathbf{F}}\mathbf{T}_{\mathbf{C}} \cdot \mathbf{t}$$

i.e.,

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} l\cos(\theta) \\ l\sin(\theta) \\ 0 \\ 1 \end{bmatrix}$$

From the above equation, we can derive two equations,

$$a\cos(\theta) + b\sin(\theta) + c = 0 \tag{5.1}$$

$$a'\cos(\theta) + b'\sin(\theta) + c' = 0 \tag{5.2}$$

where

$$a = l(\alpha_u r_{11} - (u_i - u_c)r_{31})$$

$$b = l(\alpha_u r_{12} - (u_i - u_c)r_{32})$$
$$c = \alpha_u t_x - (u_i - u_c)t_z$$
$$a' = l(\alpha_v r_{21} - (v_i - v_c)r_{31})$$
$$b' = l(\alpha_v r_{22} - (v_i - v_c)r_{32})$$
$$c' = \alpha_v t_y - (v_i - v_c)t_z$$

Note that a, b and c are functions of the horizontal camera factors, i.e., α_u, u_i, u_c, and a', b' and c' are functions of the vertical camera factors, i.e., α_v, v_i, v_c.

By solving these equations, we can find the joint angle θ. If we assume $\cos(\theta)$ and $\sin(\theta)$ to be independent variables, we solve Equations (5.1) and (5.2) by expressing them in matrix form as follows:

$$\begin{bmatrix} a & b \\ a' & b' \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} -c \\ -c' \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} a & b \\ a' & b' \end{bmatrix}^{-1} \cdot \begin{bmatrix} -c \\ -c' \end{bmatrix}$$

$$\theta = tan^{-1}(\frac{\sin(\theta)}{\cos(\theta)})$$

It is possible to obtain θ by solving one of the two equations (5.1, 5.2), and the closed-form solution is given as from Equation (5.1)

$$\theta = \pm \arccos\left(\frac{-c}{\sqrt{a^2 + b^2}}\right) + atan2(b, a) \tag{5.3}$$

or from Equation (5.2)

$$\theta = \pm \arccos\left(\frac{-c'}{\sqrt{a'^2 + b'^2}}\right) + atan2(b', a') \tag{5.4}$$

5.4 The hand pose algorithm: Strategy 2

We can, of course, simultaneously recover the palm pose and the joint angles using the perspective equations associated with each image point. However, doing so leads to a non-linear system, which is not our goal. Thus, in Strategy 2, we developed the following heuristic which combines a real-time object pose algorithm with the

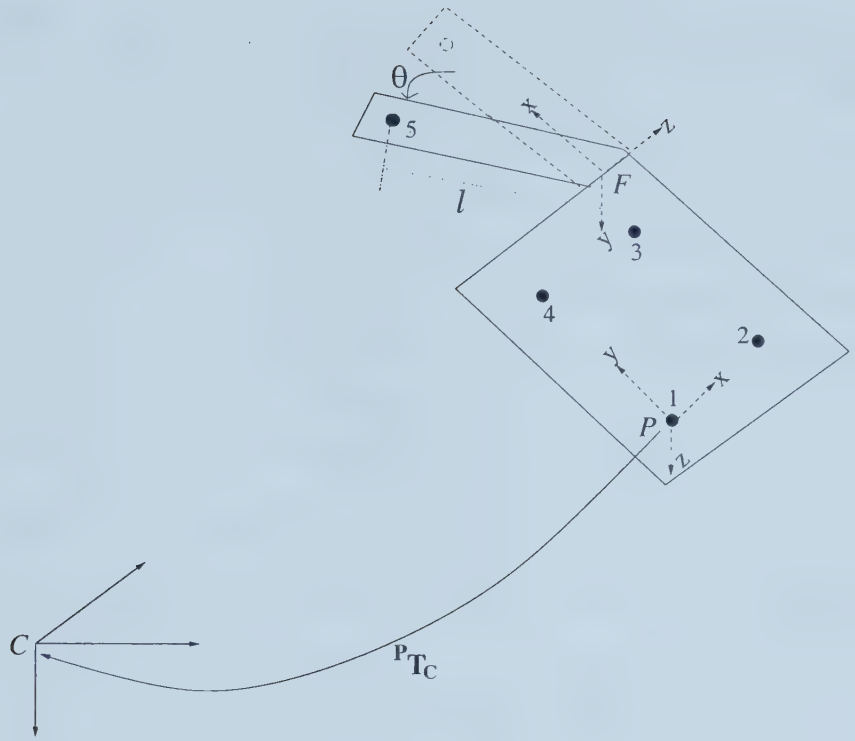


Figure 5.5: A setup for Strategy 2. All five points labeled 1 through 5 should be expressed in the palm frame P in Strategy 2. F and C are the finger frame and the camera frame respectively.

minimization of a 1D function. Strategy 2 is expected to run more slowly than Strategy 1 because of its need to run an object pose algorithm multiple times.

First, consider the problem of finding the 3D object pose $^{P}\mathbf{T_C}$ of the model shown in Figure 5.5 using all *five* points labeled 1 through 5. To find the object pose of the model, we need to know the 3D coordinates of the five points with respect to the object frame, i.e., palm frame P and the actual pixel coordinates corresponding to these object points. The pixel coordinates are obtained from the image of the model. The problem is that we do not know the object coordinates of feature 5 since we do not know the angle of the joint (recall that the joint has only one DOF). To resolve this problem, the angle is set to an initial value that will be updated using a specific criterion. Using this initial estimate, we can determine the object coordinates of feature 5 with respect to the palm frame P. Now, we can determine the 3D pose of the model, $^{P}\mathbf{T_C}$, using the five object points and the corresponding pixel coordinates. However, the 3D pose may be inaccurate because the initial joint angle may not be the actual joint angle. Once the 3D pose is found, we can back-project object points onto the image plane to verify the pose. It is easy

to see whether the pose is accurate by comparing the projected image points and the actual image points. We have to repeat the process with another angle value until an accurate pose is found. When the accurate pose is found, it means that the angle estimated is the actual angle of the finger.

Recall the following notations: (u_i, v_i) are the pixel coordinates of an object point (X_i, Y_i, Z_i), $i \in \{1 \ldots 5\}$. ${}^{\mathbf{F}}\mathbf{T}_{\mathbf{P}}$ is the transformation from the finger frame to the palm frame. $\mathbf{M}$ is the camera projection matrix. l is the distance of feature point 5 from the origin of the finger frame F.

Our algorithm can be summarized as follows:

1. Initialize the joint angle θ to a lower bound.

2. Compute the object coordinate of feature 5 with respect to the palm frame:

$$\begin{bmatrix} X_5 \\ Y_5 \\ Z_5 \\ 1 \end{bmatrix} = {}^{\mathbf{F}}\mathbf{T}_{\mathbf{P}} \begin{bmatrix} l\cos(\theta) \\ l\sin(\theta) \\ 0 \\ 1 \end{bmatrix}$$

3. Using an object pose algorithm, compute ${}^{\mathbf{P}}\mathbf{T}_{\mathbf{C}}$, which is the transformation from the palm frame to the camera frame.

4. Back-project the object points using the transformation found, ${}^{\mathbf{P}}\mathbf{T}_{\mathbf{C}}$, and the camera projection matrix $\mathbf{M}$:

$$(u_i', v_i', 1)^T = \mathbf{M} \cdot {}^{\mathbf{P}}\mathbf{T}_{\mathbf{C}} \cdot (X_i, Y_i, Z_i, 1)^T$$

5. Compute the following residual error:

$$e = \frac{\sum_{i=0}^{N-1} E_i}{N} = \frac{\sum_{0}^{4} E_i}{5}$$

   where E_i is the Euclidean distance between the projected image point (u_i', v_i') of an object point and its actual image point (u_i, v_i).

6. Save the angle and residual error pair, (θ, e).

7. If the angle θ is greater than an upper bound, stop. Otherwise, increment θ by $\Delta\theta$ and go to Step 2.

8. Find the angle which corresponds to the least residual error. This is the estimated joint angle.

There are, however, several concerns with the algorithm. First, the algorithm will be slow since we have to run an object pose algorithm (Step 3) a number of times to compute a joint angle. This weakness can be overcome by using a fast object pose algorithm and by reducing the search range between a lower and an upper bound. If we know the initial angle, assuming that the finger joint has not moved much because of a high frame rate, the next angle is close to the previous one. Therefore, the search range can be narrowed. Second, even if the projected image points and actual image points coincide and so is no residual error, it does not always mean the estimated joint angle is correct. Figure 5.6 shows a degenerate case in which two solutions are possible. When the plane in which the finger moves passes through the center of projection, there are two possible angles associated with each line of sight. This kind of problem can be overcome using a small search range because, usually, there is a large interval between the two possible angles. Also, the physiology of the finger can be considered to exclude unreasonable angles.

By repeating the above algorithm, all the other joint angles in a real hand can be computed in the same way, although computation of a middle joint angle requires six points. Note that palm pose can be either derived from the above algorithm or derived from the four coplanar points attached to the palm.

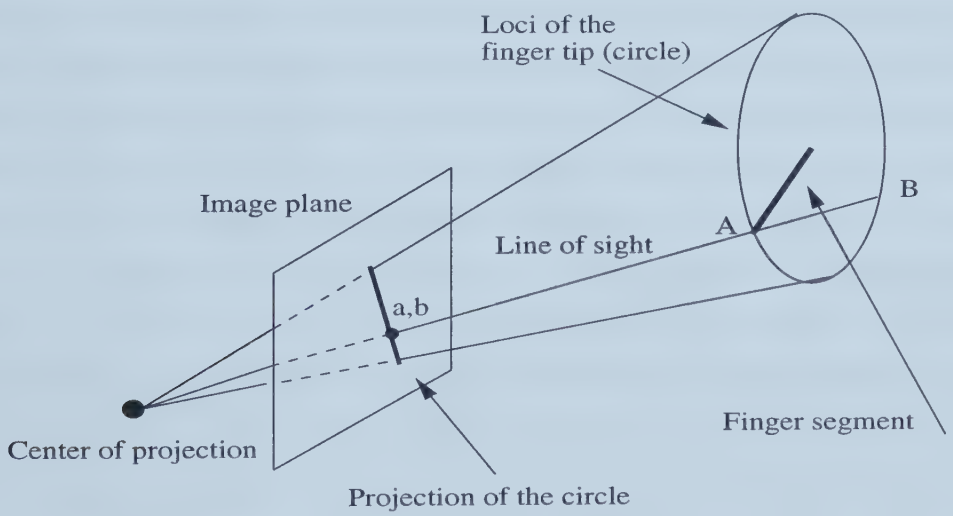


Figure 5.6: A degenerate case where the plane in which the finger moves passes through the center of projection. In this case, the trajectory of the finger tip– a circle– will be projected onto a straight line. Thus, there are two possible angles associated with each line of sight. Taken from [28] with permission.

Chapter 6

Experiments

In this chapter, we describe our experiments on synthetic and real images with Strategies 1 and 2. Performance of our algorithms is characterized by experiments on synthetic images and validated by experiments on real images of a mock-up hand. The methods of our experiments are based on those of Oberkampf *et al.* [29] for the evaluation of their weak perspective object pose algorithm.

6.1 Experiments on synthetic images

In this section, we evaluate joint angle errors of Strategies 1 and 2. We consider a hand model with one finger segment, at six distance ratios from the camera. The distance ratio is defined as the ratio of the distance from the camera to the object, over the size of the object. Synthetic images are obtained using a number of azimuth and elevation angles for the camera and two levels of image noise. Joint angles are determined by Strategies 1 and 2 from these images and compared with an actual joint angle, and errors in the joint angles are plotted against the elevation angles of the camera for various distances from the camera to the object along the optical axis. The weak perspective pose algorithm (POSIT) mentioned in Chapter 4 is used as an object pose algorithm required for Strategies 1 and 2.

6.1.1 Hand model

The hand model used for the testing has one finger segment (see Figure 6.1). Four points are placed on the palm and one point on the finger tip. The hand model should be at some distance from the camera for the weak perspective object pose algorithm to converge. The model size used to compute the distance ratio is 65 mm for both Strategies 1 and 2. All the measurements are in millimeters.

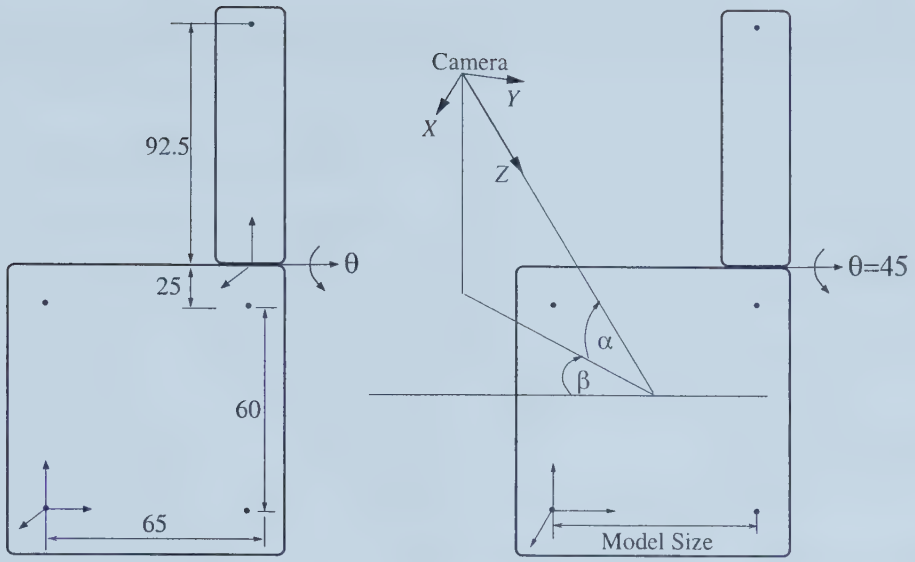


Figure 6.1: The hand model (left) and parameters (right) used for the performance testing. The measurements of the hand model are in millimeters.

6.1.2 Image generation

We produce synthetic images by perspective projection with the horizontal and vertical camera scale factors 1278.6 and 1659.5 respectively. The center of the image is assumed (256, 240). Two levels of noise in the images are considered. At noise level 1, noise lies in the range of [-0.5, 0.5] pixels. At noise level 2, it lies in the range of [-1, 1] pixels. These were generated by a uniform random number generator. Image noise is added to all of the five points on the hand model including the finger tip. Since noise is introduced in the image space, it represents a larger error in the spatial domain for a larger distance ratio.

6.1.3 Camera poses

The optical axis of the camera always points toward a point near the origin of the palm frame because we fixed the translation in x and y directions as -32.5 mm and 30 mm respectively. The camera is successively located at six distance ratios: 4, 7, 10, 15, 20 and 30. For each of these distance ratios, 17 elevation angles, from 10 to 90 degrees, are considered in synthetic data. These camera elevation angles are denoted by α in Figure 6.1. For the elevation angle of 90 degrees, the camera is at nadir. All the joint angle errors are plotted against these elevation angles. β in Figure 6.1 represents camera azimuth angle. We take the average of the joint angle errors for 72 camera azimuths, in increments of five degrees. We plot these

average errors against 17 elevation angles for the two noise levels and the six camera distance ratios.

6.1.4 The joint angle error

The joint angle error is the difference between a joint angle of the hand model used to produce the synthetic images and a computed joint angle. We fixed the joint angle of the model hand at 45 degrees. Hence, the joint angle error is the difference between 45 degrees and the computed joint angle. As shown in 5.3.1 in Chapter 5, for Strategy 1, the joint angle can be determined from the matrix inverse approach or from Equations (5.3) and (5.4). Since there are four possible answers from Equations (5.3) and (5.4), we choose the closest one to the correct angle for performance evaluation. For Strategy 2, we select the closer one to the correct angle when there are two minima with similar values.

6.1.5 Analysis of the error plots

Results from the experiments are shown in Figures 6.2 and 6.3. These plots show that both Strategies 1 and 2 are more sensitive to image noise when the camera is close to nadir or at nadir. The reason for this sensitivity is that the palm pose can not be computed accurately enough at nadir. It is known that object pose is the least accurately computed when the camera is at nadir to the coplanar points. Oberkampf *et al.* [29] showed that nadir configuration was the most sensitive to image noise and the largest errors occurred in object pose computations. The reason is obvious in that "all the rotations around axes in the plane of the object displace the feature points in directions which are close to the directions of the lines of the sight, and are difficult to detect because they produce few changes in the image." [29].

In Strategy 1, choosing one of the four solutions from Equations (5.3) and (5.4) performs much better than the matrix inverse approach. The reason for the poor performance of the matrix inverse approach may be that the joint angle is computed using all the coefficients including those with big errors caused by the inaccurate palm pose. However, selecting the best answer from Equations (5.3) and (5.4) corresponds to using coefficients with the less errors. The performance of Strategy 2 is overall the most stable and accurate. Strategy 2 is robust because it searches all the possible range of the joint angle. As distance ratio increases, non-nadir camera poses tend to get sensitive to image noise for both strategies.

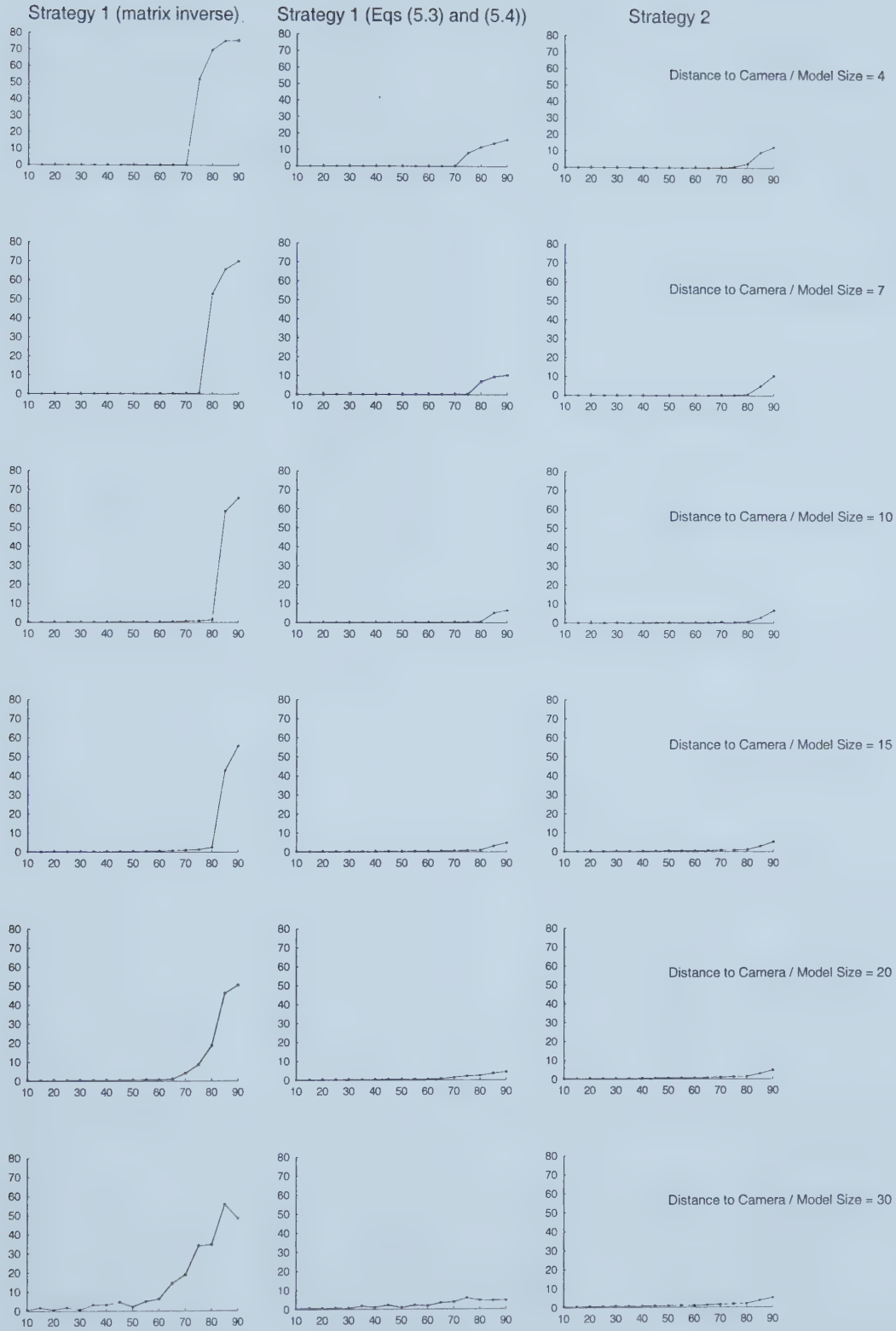


Figure 6.2: Tests with noise level one, i.e., image noise of one-pixel amplitude. The horizontal axis represents elevation angle of the camera. At 90 elevation angle, the camera is at nadir. The vertical axis represents average joint angle error of 72 azimuth for each elevation angle.

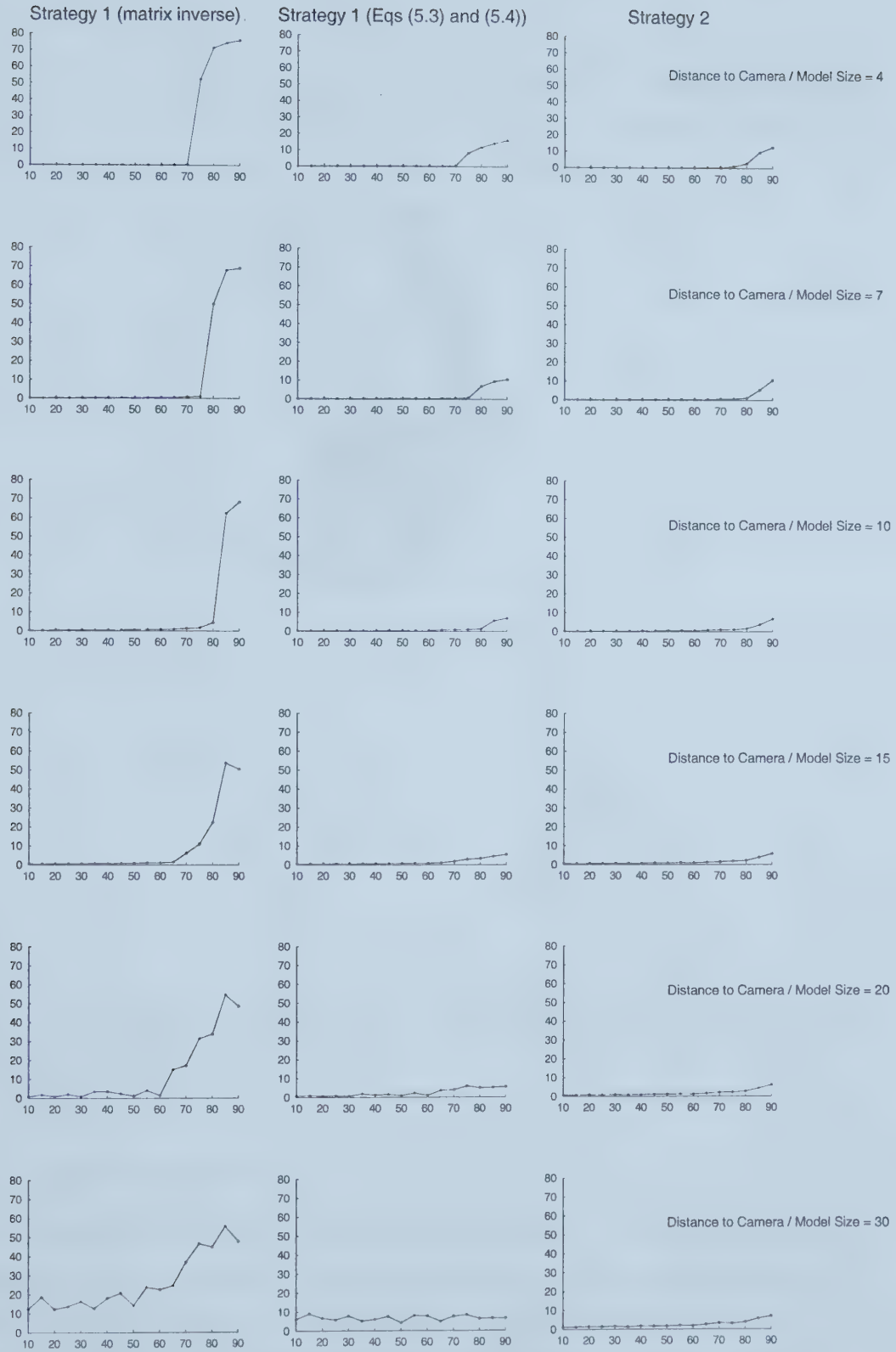


Figure 6.3: Test results with noise level 2, i.e., image noise of two-pixel amplitude. The horizontal and vertical axes represent the camera elevation angle and average joint angle error respectively.

6.2 Experiments on real images

We present results with real images of our mock-up cardboard hand (Figure 6.4) at different distance ratios and different viewing angles. Real experiments validate the performance characterization from the theoretical experiments.

Figure 6.4: A mock-up hand made of cardboard with only one finger segment. The joint angle can be measured with a protractor.

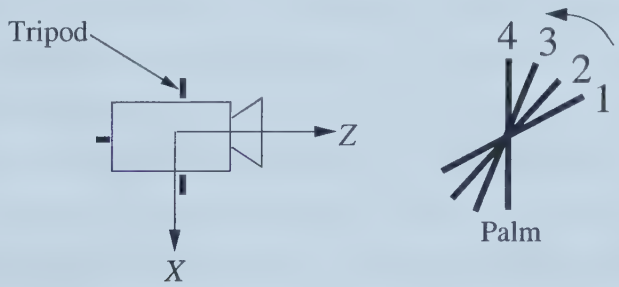


Figure 6.5: View from the ceiling. At Step 1, the camera is at a much slanted angle to the target. At Step 4, the camera is at nadir. The biggest errors in joint angles are expected at Step 4.

6.2.1 Experimental setup

The cardboard hand with one finger segment was designed to verify the algorithms' performance with real images. The dimensions of the mock-up hand is equal to those of the hand model (Figure 6.1) used in the theoretical experiments. This mock-up hand gives more rigidity than the human palm so that the markers on it do not move independently of each other. The joint angle of the finger can be easily measured using a protractor.

As shown in Figure 6.5, the mock-up hand, not the camera, is rotated in place in four steps, becoming nadir at Step 4. Sample images taken at each step at distance ratio of 12 are shown in Figure 6.6. This four-step sequence is repeated five times for each distance ratio. We plot the averages of the joint angle errors against each step for each distance ratio. The mock-up hand is positioned at four distance ratios[1]: 12, 15, 20 and 30. The joint angle was fixed throughout the tests at 45 degrees.

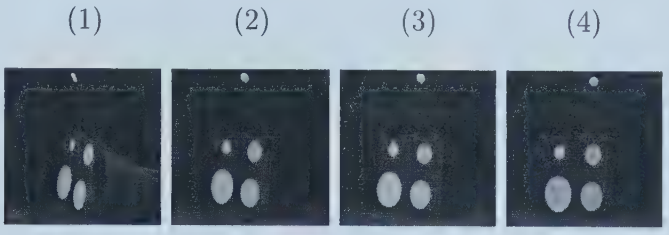


Figure 6.6: Four sample images representing the four configurations in the experiments. Each number above each image indicates each step in Figure 6.5.

6.2.2 Test results

The biggest errors occurred at nadir, i.e., at Step 4 for both Strategies 1 and 2 as shown in Figure 6.7. The reason of this sensitivity at nadir is obvious as mentioned in Section 6.1.5. At distance ratios of 20 and 30, non-nadir configuration (Step 3) started to show the sensitivity to image noise. This agrees with the results from the experiments on synthetic images. Strategy 1 with Equations (5.3) and (5.4) and Strategy 2 have roughly the same performance, although in theoretical experiments, Strategy 2 showed slightly more robustness with bigger distance ratios.

6.3 The dual solution problem in Strategy 2

In this section, we discuss the possibility of a dual solution in Strategy 2 with real images. In Strategy 2, we have to back-project all the object points to the image plane to compute the residual error, as we explained in Section 5.4. All the object points are back-projected at each angle in the search range, and each object point makes a trajectory on the image plane.

Two example images with the trajectories of the projected object points are shown in Figure 6.8. Their residual error plots are shown right below each image. Usually, the trajectories cross the centroids of the circular markers at a certain angle

[1]Because of the very narrow field of view of the camera we used, smaller distance ratios than 12 were not possible.

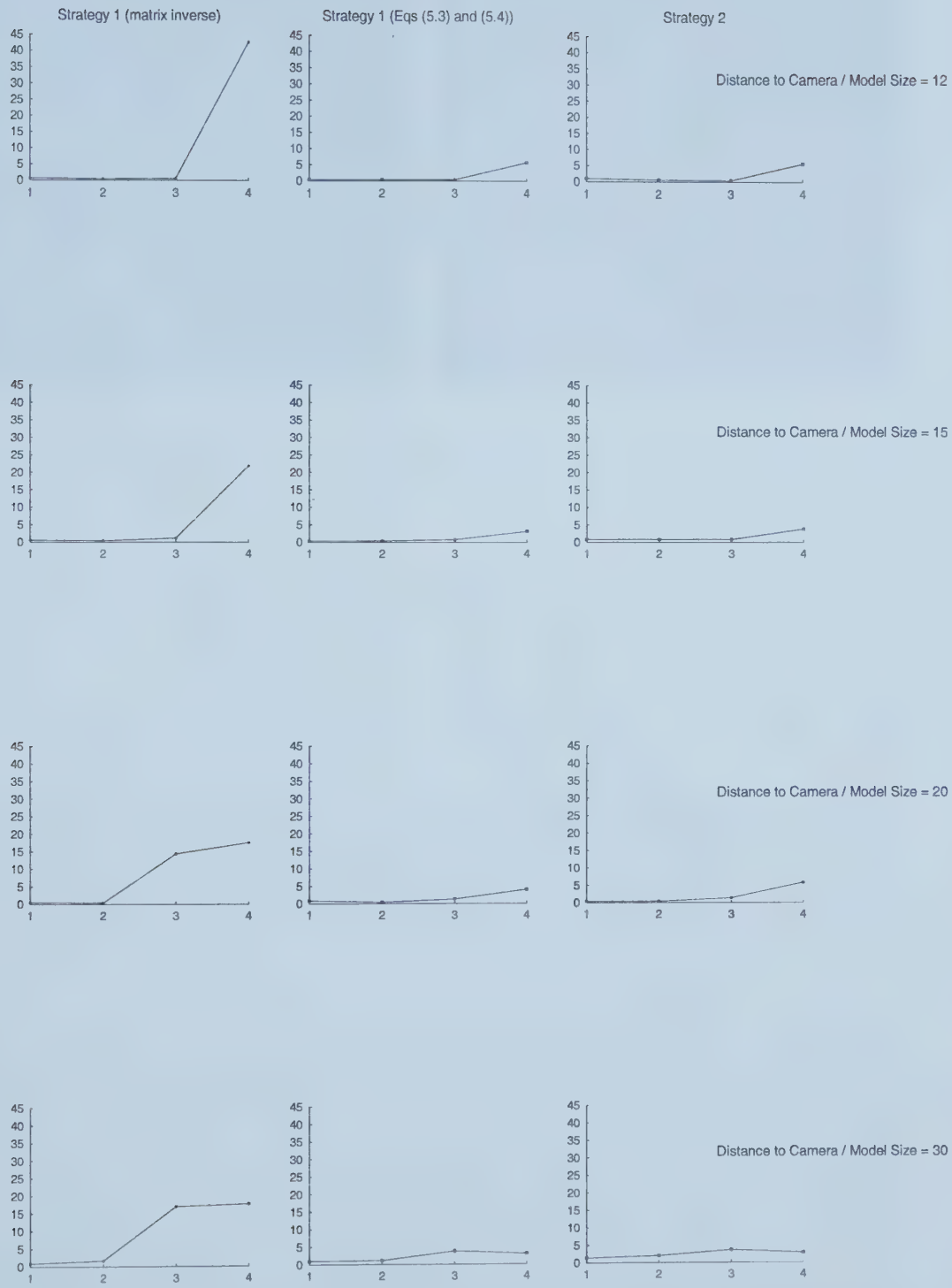


Figure 6.7: Experimental results with the real images of the mock-up hand at various distance ratios. The results agree with the theoretical experiments. The vertical axis represents the average joint angle error.

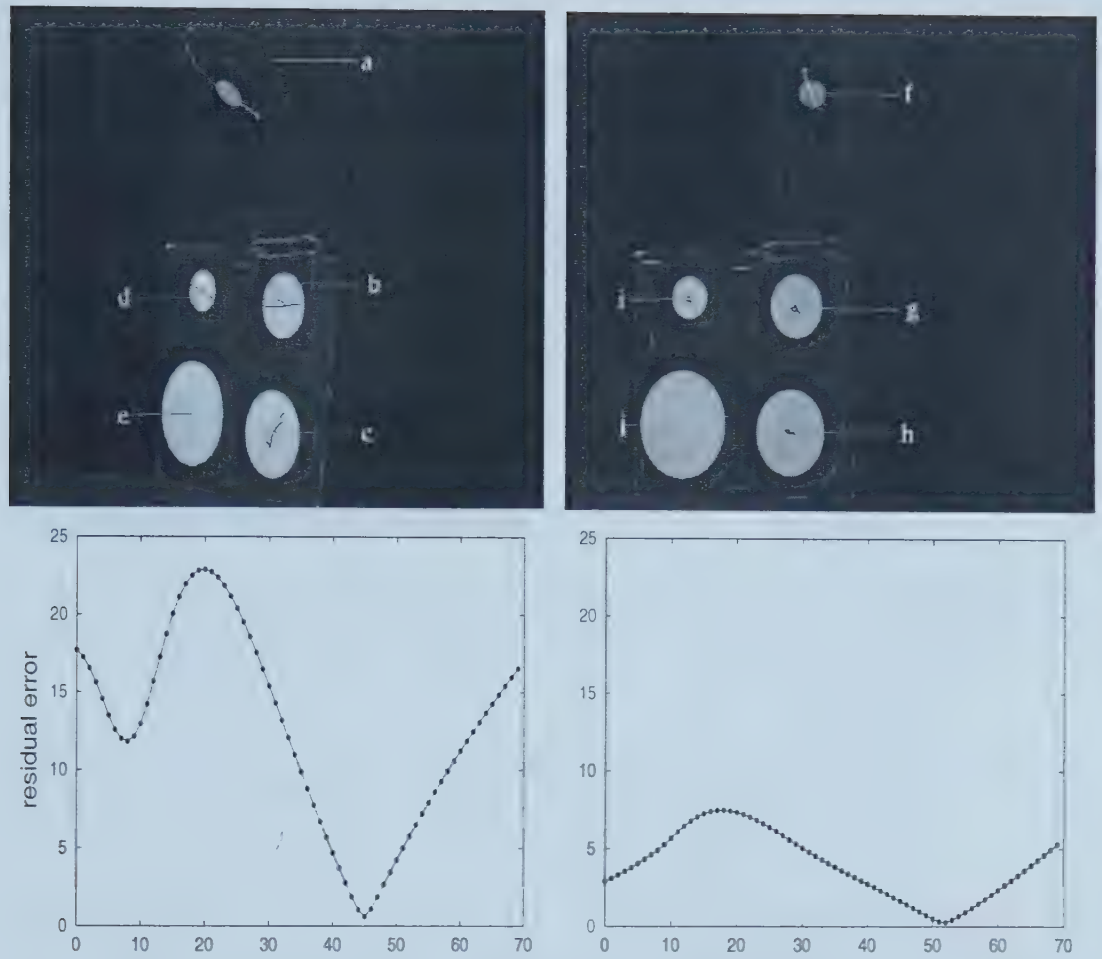
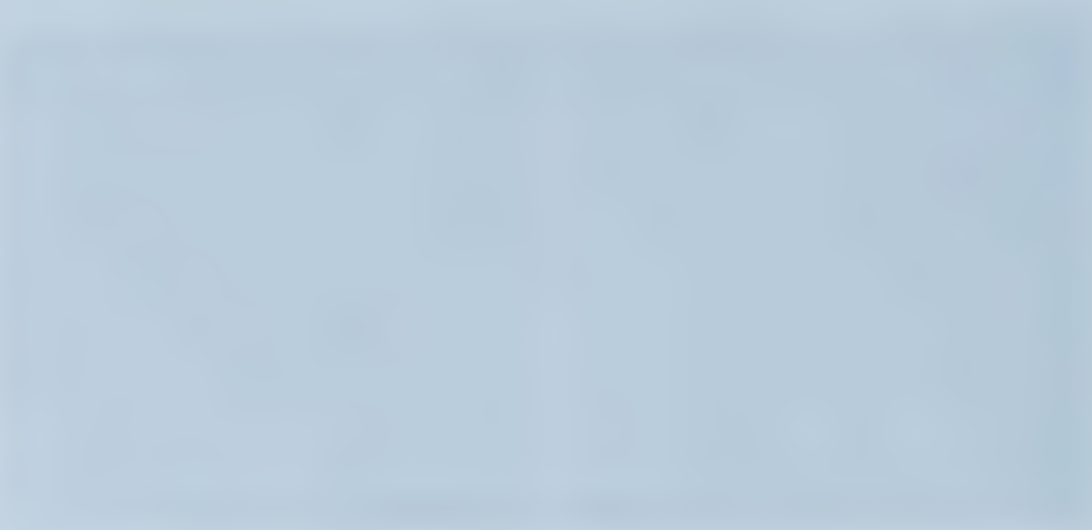


Figure 6.8: Trajectories (a $\sim$ j) over the circular markers are generated by rotating the finger from zero to 70 degrees (horizontal axis) and by back-projecting all the object points at each angle. Their corresponding residual error plots are shown right below. The actual measured joint angle of the mock-up hand shown in the pictures is 45 degrees.

of the finger rotation, suggesting the angle with the minimum residual error as the actual joint angle. When the finger moves normal to the optical axis, the trajectory of the finger is big (Trajectory a), and the slope of the residual error is steep (left residual error plot of Figure 6.8). However, when the finger moves along the optical axis, the trajectory is small (Trajectory f), and the slope of the residual errors is blunt (right residual error plot of Figure 6.8).

Usually, there exists one global minimum with a small residual error whether the finger moves along the optical axis or not. The degenerate configuration discussed in Chapter 5 can produce two minima with the same residual errors theoretically. In reality, two minima with the exactly same residual errors do not occur due to



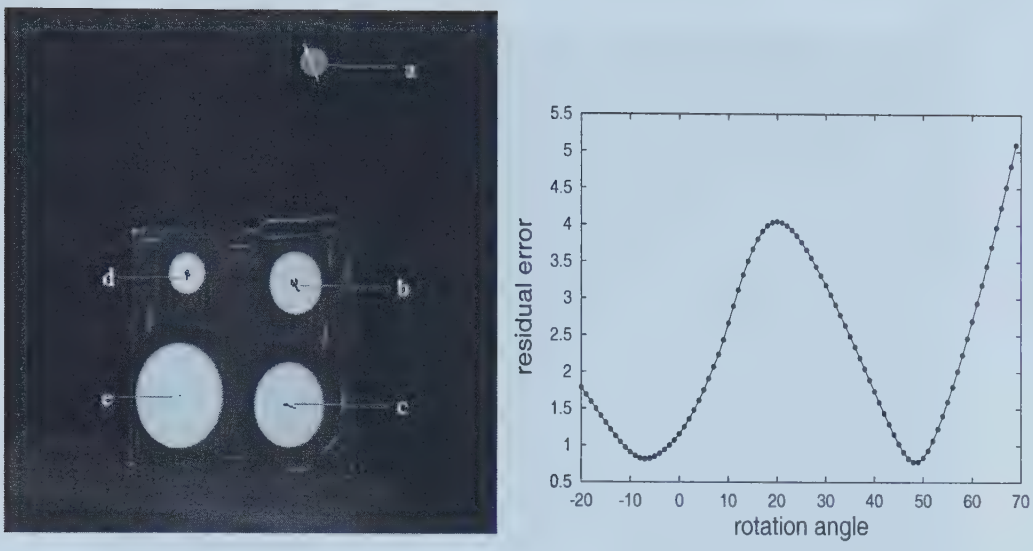


Figure 6.9: Left: The finger trajectory (Trajectory a) is almost a straight line. The trajectory comes back to pass very near the centroid again. Right: Residual errors with two minima with similar values.

noise and granularity[2] of the rotation angle in the heuristic search. The degenerate configuration will yield two minima with the similar residual errors in reality. Figure 6.9 shows such a case where there are two minima with similar residual errors at -7 and 48 degrees. The residual errors are 0.815 and 0.776 respectively. Trajectory a over the finger marker is almost a straight line, and it passes near the centroid of the finger marker twice during the search range of -20 to 70 degrees. When there are two angles with similar residual errors, it's possible that the angle less close to the correct angle can be chosen because it can have slightly smaller residual error than the one more close to the correct angle due to noise. It is not the case in Figure 6.9 because the actual measured angle is 45 degrees and the smaller residual error occurs at 48 degrees, not at -7 degrees. We can verify that both poses computed with -7 and 48 degrees were accurate by back-projecting the model points to the image plane. When the centroids of the markers on the model are projected onto the image plane with the joint angles of -7 and 48 respectively, all the projected points lie very near the centroids of the markers in the images as shown in Figure 6.10, which indicates that the hand poses found were fairly accurate for both joint angles. This case with the similar residual errors supports our analysis that there can be a dual solution when the plane in which the finger moves passes through the center of projection.

[2]We rotated 1 degree each time.

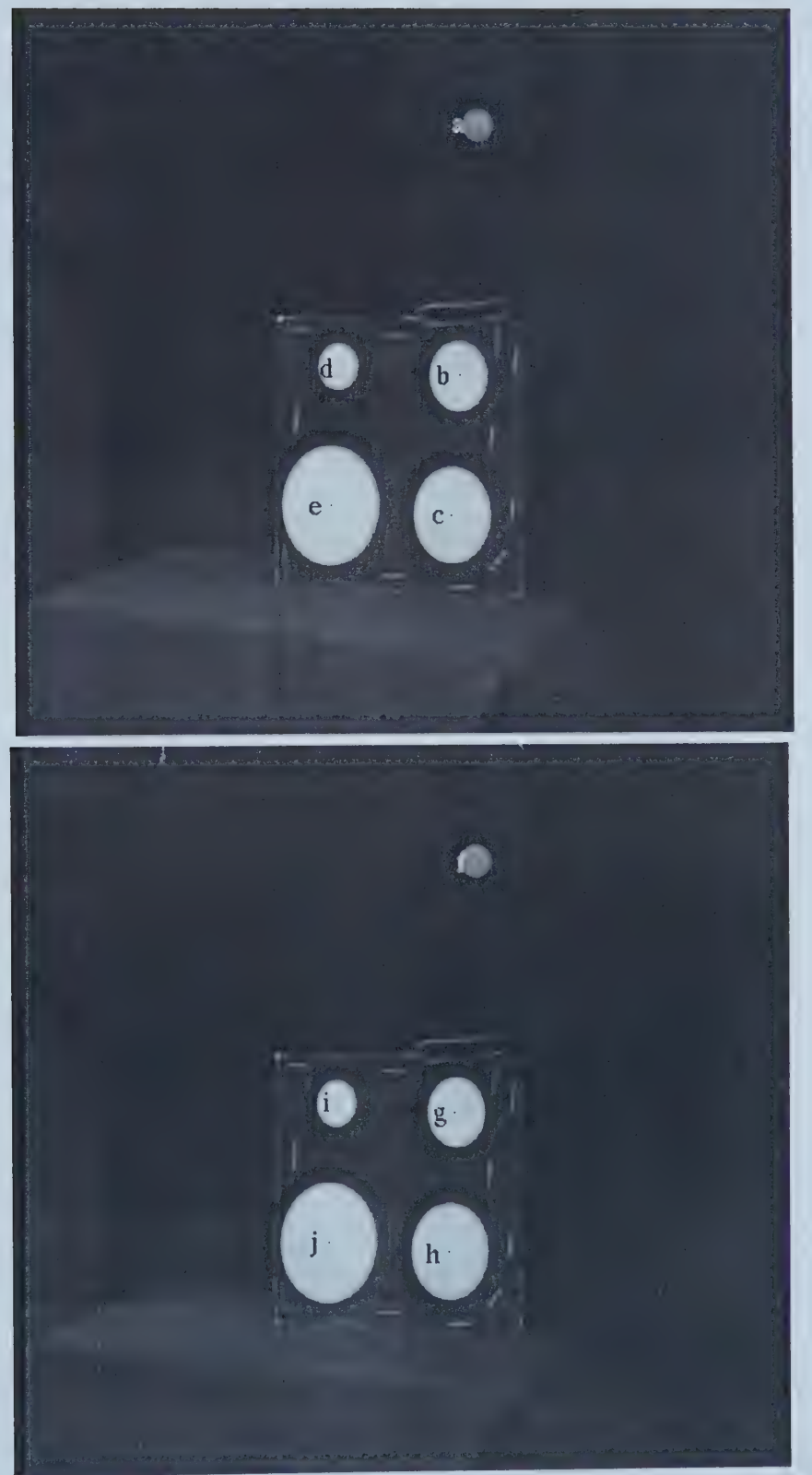


Figure 6.10: The top image shows the projections (a $\sim$ e) of the object points when the finger joint angle is assumed -7 degrees, and, below, the projections (f $\sim$ j) with 48 degrees.

6.4 Conclusion from experiments

We conclude that despite the sensitivity at nadir, performance of Strategy 1 (when we select the best angle from the four solutions) and Strategy 2 is good, considering the average errors are usually within five degrees with real images. Experiments on synthetic and real images showed that both strategies are more sensitive to image noise at nadir than at non-nadir camera poses. This is due to the sensitivity of the object pose sub-algorithm at nadir, and this can not be avoided when a pose is computed from a single image obtained by a CCD camera [1].

Chapter 7

The complete system

A complete hand-tracking system was built so that a real human hand could be tracked in 3D space. Both Strategy 1 (choosing one among the four angles) and Strategy 2 were tried to track a real hand wearing our special glove. We describe the details of the complete hand-tracking system in this chapter.

7.1 The glove

To support our assumptions (the fingers move in a plane and the palm is rigid) on a real hand, we developed a special glove (Figure 7.1). The process of making it is shown in Figure 7.2. For fingers, thin wood board segments are joined together using small hinges, which make each finger rotate in a plane and disable adduction and abduction. These joined wood board segments are attached to a wood board palm. The wood board palm removes all the flexibility caused by non-rigidity of the human palm. This hand-like structure is attached to a black glove. Markers are printed

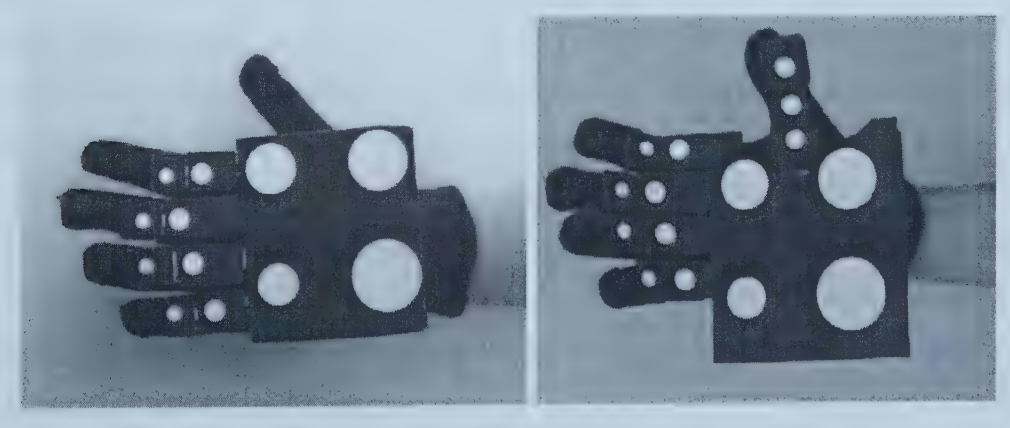


Figure 7.1: The special glove to be worn by a user. The right image shows the complete glove with the thumb considered.

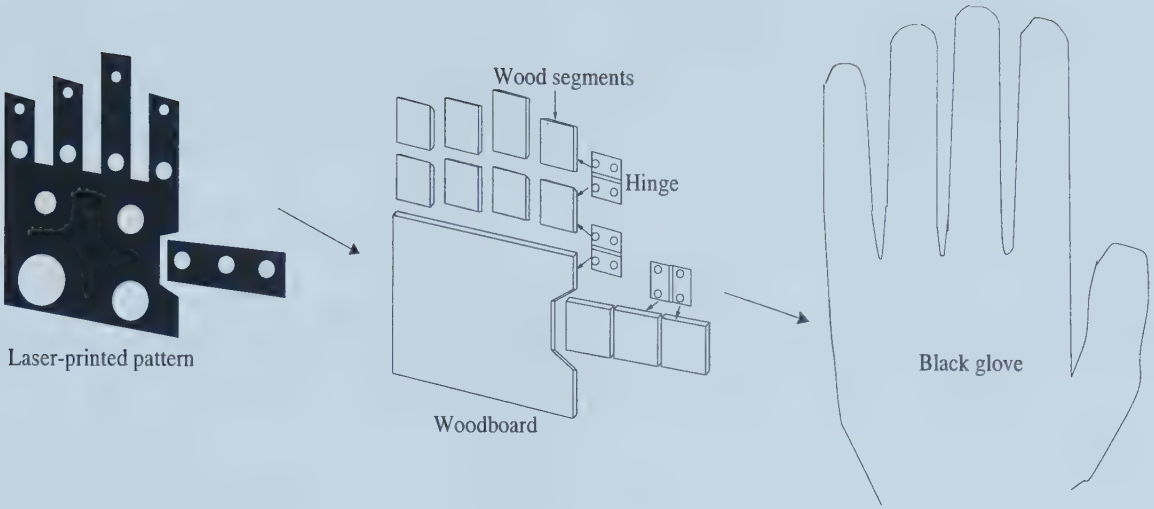


Figure 7.2: The process of making the glove. A laser-printed hand pattern is glued to a wood structure. Then, it is attached to a black cotton glove. Small hinges force fingers to move in a plane.

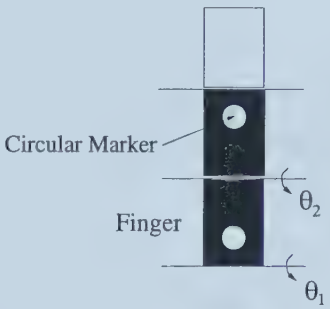


Figure 7.3: One finger. Circular markers are positioned between joints to prevent deformation.

and glued on the palm and the finger segments. Note that circular markers are positioned between finger joints, not onto joints, to prevent deformation of markers when the fingers bend (Figure 7.3). We tried to reduce the invasiveness of the glove by using a cotton glove and thin wood board segments. The thumb currently has three DOF's (Figure 7.4).

The advantages of using such a supporting structure are two fold. First, doing so helps to acquire more accurate marker positions, leading to accurate computation of joint angles. Second, the glove will fit almost all the adult subjects, requiring no calibration of the system, which would be necessary if a naked hand were used since each person's hand is different in size.

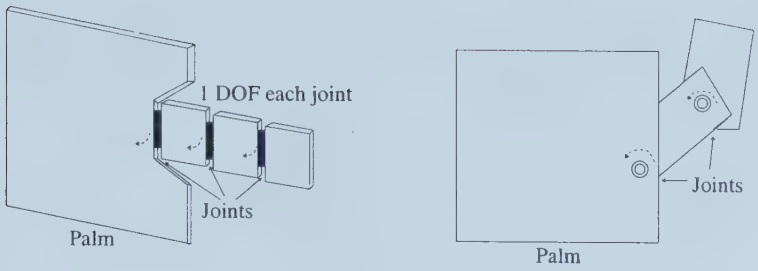


Figure 7.4: The left figure is the current thumb model, and the right is another suggested model where the finger rotates in a plane parallel to the palm. The right one gave more comfort when they were implemented with a wood structure.

7.2 The object pose algorithm in our system

Perspective-n-Point pose algorithms are important for determining joint angles in our strategies. In principle, any PnP object pose algorithm can be used in our strategies. However, since real-time performance of hand tracking is the most desirable, the faster an object pose algorithm is, the better for our system. The algorithm ([1], [29]) employed for our system is reasonably fast as the authors developed it to provide 60 pose computations per second in a video-based 3D mouse system on a personal computer.

The object pose algorithm deals with two distinct cases: coplanar and non-coplanar. However, in our system with Strategy 2, the configuration of points can be coplanar or non-coplanar depending on the joint angles of the moving fingers[1]. In Figure 7.5, if the finger joint angle is zero, all five points lie on a plane. We found out that when the joint angle was small (approximately less than 20 degrees), regarding the configuration either as coplanar or as non-coplanar did not return very accurate pose. To make such a configuration always non-coplanar so that we do not have to worry about the case when a joint angle is small, we use four *imaginary* points (Figure 7.5). Also, we need to compute pixel coordinates corresponding to the four imaginary object points. Since we can compute transformation from the palm frame to the camera using the four actual coplanar points on the palm, we can back-project *imaginary* object points using the transformation found from the four coplanar points to get the corresponding *imaginary* pixel coordinates required by the algorithm. With these four imaginary object points, model configurations always become non-coplanar without regard to the angles of the finger joints.

[1]This is not a problem with Strategy 1 since the pose algorithm uses only four coplanar points on the palm.

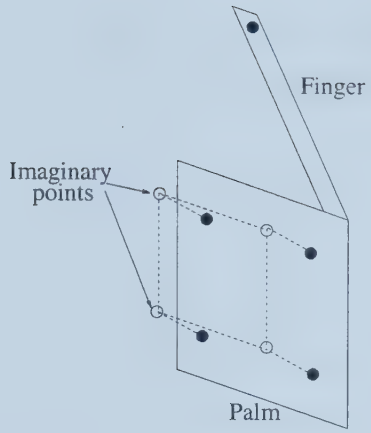


Figure 7.5: For Strategy 2, four virtual object points are added to the palm in order to make the hand model non-coplanar regardless of the joint angle values.

7.3 Feature extraction and correspondence

Accurate extraction of image points of markers and establishing correspondence between markers and their image points are basic requirements of our system. To facilitate this process, black background and white circular markers were used. Note that the use of circular dots as markers considerably reduces the image processing costs and allows for a very accurate localization of image points.

For the centroid extraction of the markers, we segment the captured image using thresholding to get its binary image (e.g., third column of Figure 7.6). Then, each pixel is scanned. If it is a white pixel, some pixels above and below the white pixel are examined to see if the white pixel belongs to an already labeled marker or to a new marker and labeled accordingly. A centroid is the sum of the positions of all pixels in a labeled marker divided by the number of the pixels.

We use circular dots of different sizes for the palm markers. We can establish correspondence between the palm markers and their image points just by counting the number of pixels in each extracted marker. This can be problematic when the markers are seen at a much slanted angle and the markers are deformed. Therefore, sufficiently different sizes should be allowed. For the fingers, the bottom joints have circular dots of the same size, which are bigger than the middle segment dots. To discern among dots of the same size on the glove, hand geometry is used. This will limit a subject's hand motion but allows enough motion to test the algorithm.

The speed of the system can be different depending on which types of markers are used and how they are extracted. Different options for the markers are color

blobs or LEDs (light emitting diodes). We concentrated on the algorithm rather than investigating different options for the markers and their extraction.

7.4 The system in action

7.4.1 System setup

A gray scale NEC T1-23A CCD camera was set up about one meter away from a subject's hand. For easy verification of our algorithm, we developed a graphic hand simulator with OpenGL to animate the motion of the real hand wearing the special glove. The machine we used was a Pentium III 500 MHz PC running Linux.

7.4.2 In action

The graphic hand model was successfully following the poses of the real hand wearing the glove either with Strategy 1 or with Strategy 2. With Strategy 1, the system could determine hand poses at up to 12 Hz[2]. We select one of the four possible answers by back-projecting the four answers and by choosing the closest one to the actual image point of the finger marker. Strategy 2 was slower than Strategy 1. At the beginning, the whole search space, i.e., the possible range of the finger flexion[3], is scanned. After that, we can narrow down the search space assuming that the next joint angles are not much different from the current ones. We found that the search range of a joint angle could be ± 5 degrees from the previous angle. Hand poses are determined at up to 8 Hz with Strategy 2. Therefore, the joints can rotate $\pm 40°/s$. The increment step of the search angle was 1 degree. Although Strategy 1 was faster than Strategy 2, Strategy 1 was more unstable, i.e., unreasonable joint angles were more frequent in the poses of the hand simulator.

It is difficult to get the ground-truth for the joint angles of the moving fingers of a real hand. Although the ground-truth for the fingers' motion is not available, one can evaluate the performance by comparing the real motion and the one performed by the graphic model. Figure 7.6 shows sample images of a real hand moving in 3D space wearing the glove, the animated hand model following the motion of the real hand, and the tracked real hand.

[2]Note that display time of graphic hand model was not included.
[3]We assume it is [-10, 90] degrees.

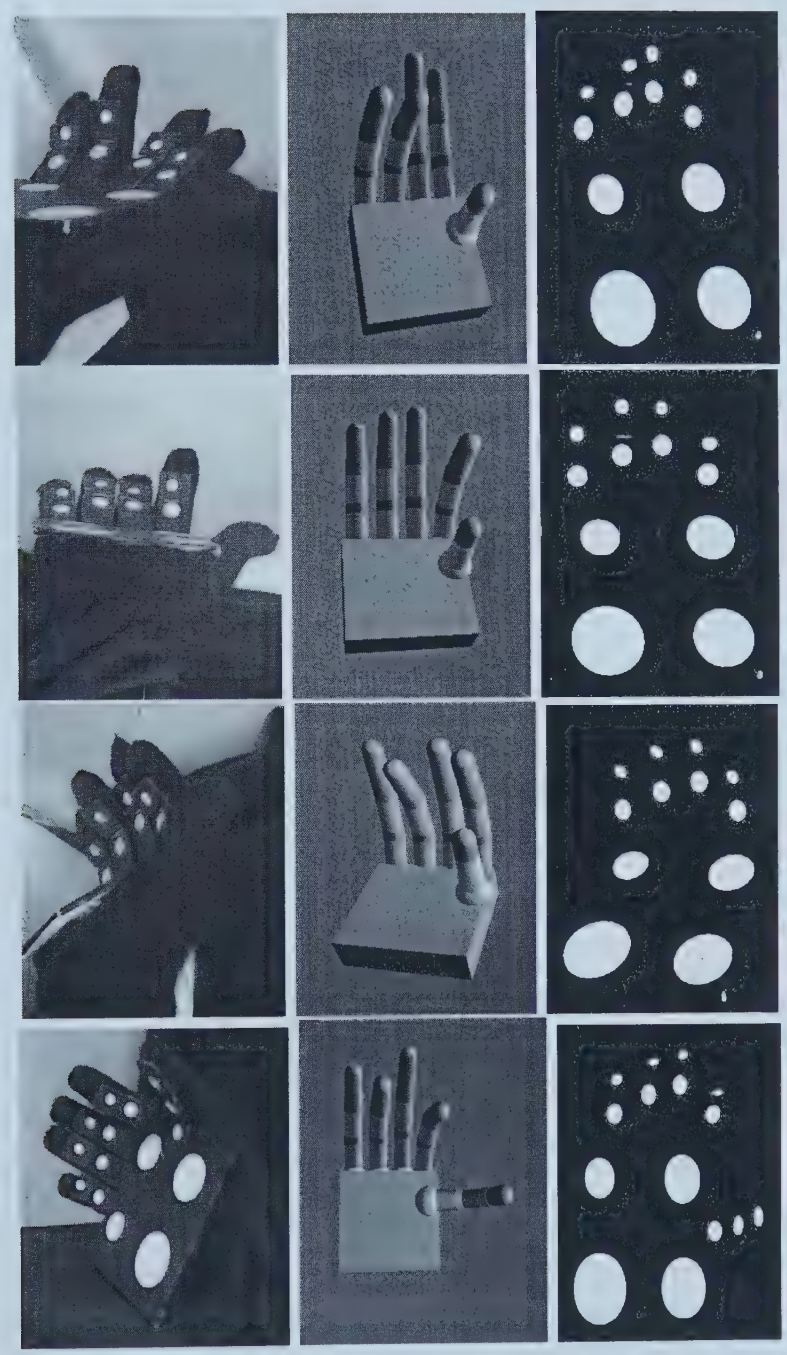


Figure 7.6: Left: A real hand in action (images were taken using another camera from different directions). Middle: The graphic hand model following the 3D poses of the real hand (palm and fingers). Right: The tracked real hand. The last three pictures also show the thumb in action.

Chapter 8

Discussions

8.1 Summary

Real-time performance of a hand-tracking system is important since many applications require real-time execution of the system. In this thesis, we developed two algorithms suitable for a real-time hand-tracking system based on the Perspective-n-Point problem.

In Strategy 1, a set of closed-form equations is found to determine finger joint angles. The equations are derived from projective relations between a finger marker and its image point. The palm pose is determined by an existing PnP algorithm. Strategy 2 is an iterative algorithm with a 1D heuristic search. A finger joint angle is found by rotating the finger from a lower bound to an upper bound and by computing residual errors at each rotation angle.

Experiments on synthetic and real images showed that both strategies are more sensitive to image noise when the camera is at nadir. However, overall, the joint angle errors are within ten degrees.

A complete hand-tracking system was built, and both strategies were tested. The system could run in real time with either strategy, although Strategy 1 (12 Hz) is about 1.5 times faster than Strategy 2 (8 Hz). We used a special glove made of wood board segments to remove the noise caused by the independent movement of the markers that resulted from the deformation of the hand. One of its advantages is that it fits almost all subjects, and no system calibration is necessary for different subjects.

8.2 Future research

It may be possible to combine all the approaches discussed in this thesis to implement one complete system. When the camera is at a sufficiently slanted angle, the matrix inverse approach of Strategy 1 might be used, eliminating the process of choosing one of the four angles. When the camera is at nadir, Strategy 2 might be used since it is the most stable. By testing camera angles to the object, this should be possible.

One of the biggest problems in computer vision-based approaches is the occlusion problem. Currently, we assume that all markers can be seen at any given time. However, in reality, this is not the case, and the markers are easily occluded. We are considering marker tracking algorithms to alleviate the occlusion problem. Also, different markers such as rings as in Dorner [3] are also considered since a ring around a finger segment can be seen from almost any direction. Another way to alleviate the occlusion problem is to use several cameras set up in different directions. Multiple cameras can help overcome the sensitivity to image noise at nadir by simply choosing the object pose computed from the camera at a slanted angle to the target hand.

In the current system, we used the circular markers of the same size for the finger segments, and we used the geometry of the hand to discern these markers, limiting the motion of the hand. However, the following approach would be more desirable. Once the palm pose is known, we can use the mapping from the palm plane to the image plane to project each finger marker onto the image plane. The angle value of each finger can be set to a fixed value (e.g., the middle value of the possible finger flexion), or to the computed value from the previous image. Doing so provides a predicted 2D location for each finger marker. Distinguishing between fingers can then be carried out by comparing the predicted 2D locations and the actual 2D locations.

8.3 Applications of the current system

It is difficult to use the current system in real applications such as gesture recognition and tele-operations of a robot hand because of the problems with our current system, e.g., occlusion of the markers and the requirement of a black background for marker detection. These problems are difficult to solve in the case of hand tracking based on computer vision techniques.

However, the current system may be useful for some applications as it is or

with simple modifications. One application is virtual reality or augmented reality in which it can be used as an input device. Consider that a user wearing 3D glasses wants to interact with the virtual environment on a computer screen. Information concerning 3D positions of the hand can be used to navigate the 3D environment. A few gestures can be mapped to manipulate 3D objects and interact with the virtual environment.

The current system might be applied to the CAVE. The CAVE is a display device with 3 walls onto which three dimensional computer generated images are projected, immersing the user in a 3D environment. Usually a Polhemus sensor with long cables attached is used to determine the user's position within the CAVE. This poses safety problems. Since our current system can track the 3D position and orientation of a palm accurately and some gestures can be specified with joint angles, it can replace Polhemus sensors. There are several tasks to be done for the current system to be used in the CAVE. Since the user is moving around in the CAVE, the camera should follow the user's hand wherever it goes. The CAVE is dark, so the white markers are hard to detect. Different types of markers (e.g., light emitting diodes) can be used.

With our current hand-tracking system, the most likely application would be as an input device equipped with a few gestures. For example, an interface with a few gestures can be set up to communicate with a robot. The current system can replace the mouse in computer games as Segen and Kumar [6] did, providing a more natural interface for game players.

Bibliography

[1] D.F. DeMenthon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123-141, 1995.

[2] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. *Computer Graphics*, vol. 25, Number 4, July 1991

[3] B.Dorner. Hand shape identification and tracking for sign language interpretation. *Looking at People Workshop, Chambery, France*, 1993.

[4] J.M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Proc. 3rd ECCV, Stockholm, Sweden*,volume II, pages 35-45, 1994.

[5] J. Segen and S. Kumar. Driving a 3D articulated hand model in real-time, *IMDSP'98, Alpbach, Austria* July, 1998.

[6] J. Segen and S. Kumar. GestureVR: Vision-based 3D hand interface for spatial interaction. *ACM Multimedia 98 - Electronic Proceedings.*

[7] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):201-211, 1973.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, vol.24, pp. 381-395, 1981.

[9] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33-44, July 1989.

[10] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE transactions on Robotics and Automation*, 5(2):129-142, April 1989.

[11] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robotics and Automation*, vol.3, pp. 323-344, 1987.

[12] L. Quan and Z. Lan. Linear n-point camera pose determination. *Transactions on PAMI*, vol. 21, No. 7, July 1999.

[13] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441-450, May 1991.

[14] T. Speeter. Transforming human hand motion for telemanipulation. *Presence*, vol. 1, No. 1, Winter 1992, MIT press.

[15] J. Webb and J. Aggarwal. Structure from motion of rigid and jointed objects. *Artificial Intelligence*, 19:107-130, 1973.

[16] J. Lee and T.Kunii. Constraint-based hand animation. *Models and Techniques in Computer Animation*, pp.110-127. Springer-Verlag, 1993.

[17] R. Rashid. Towards a system for the interpretation of moving light displays. *IEEE transactions on PAMI*, vol. 2, No. 6, November, 1980.

[18] J. Napier. *Hands.* Pantheon Books, New York, 1980.

[19] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision computing*, vol. 1, No. 1, pp.5-20, 1983.

[20] J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE transactions of PAMI*, vol.2, No. 6, pp.522-536, 1980.

[21] A. Downton and H. Drouet. Model-based image analysis for unconstrained upper-body motion. *International Conference on Image Processing and its Applications*, pp.274-277, 1992.

[22] R. Davis. Clinical gait analysis. *IEEE engineering in Medicine and Biology magazine*, vol.7, No. 3, pp.35-40, 1988.

[23] M. Yamamoto and K. Koshikawa. Human motion analysis based on a robot arm model. *IEEE transactions on Computer Vision and Pattern Recognition*, pp.664-665, 1991.

[24] Q. Delamarre and O. Faugeras. Finding pose of hand in video images: a stereo-based approach. *Third International Conference on Automatic Face and Gesture Recognition*, April, 1998.

[25] O. Faugeras and G. Toscani. Camera calibration for 3D computer vision. *International Workshop on Machine Vision and Machine Intelligence, Tokyo, Japan*, pp.240-247. 1987.

[26] S. Ullman. The interpretation of visual motion. *Ph.D. thesis, MIT*, 1977.

[27] W. Clocksin. Inference of structural descriptions from visual examples of motion: preliminary results. *DAI Working Paper 21, University of Edinburgh, UK*, 1977.

[28] K. Kwon, H. Zhang, and F. Dornaika. Hand pose recovery using a single camera. *International Conference on Robotics and Automation*, pp.1194-1200, May 2001.

[29] D. Oberkampf and D.F. DeMenthon, L.S. Davis. Iterative Pose Estimation using coplanar feature points. *CVGIP: Image understanding*, vol. 63, no. 3, May 1996.

[30] R. Wilson. *http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-source.html.*

[31] B. Wilburn. *http://velox.stanford.edu/~wilburn/Calibration/report.html.*

[32] *http://www.virtex.com/products/hw_products/cyberglove.html.*